Motion Analysis of 3D Rigid Curves from Monocular Image Sequences

Théodore Papadopoulo

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Motion Analysis of 3D Rigid Curves from Monocular Image Sequences

Théo Papadopoulo

N° 2779

Janvier 1996

PROGRAMME 4

ISSN 0249-6399

Motion Analysis of 3D Rigid Curves from Monocular Image Sequences

Théo Papadopoulo

Programme 4 — Robotique, image et vision

Projet Robotvis

Rapport de recherche n°2779 — Janvier 1996 — 311 pages

Abstract: This thesis deals with the difficult problem of the recovery of the motion and structure of a scene, from monocular sequences of images, in the case of a scene constituted of a single rigid curve. The interest in this problem is both theoretical and practical: indeed, even if the situation we are looking at seems simplified, it is important to have a good understanding of it as it is, in some way, the generic situation that is encountered. This is because, computing the motion from edges, engenders problems. The most famous of them is the so-called aperture problem: at an edge point, only the normal component of the 2D velocity field is recoverable. In the face of this problem, researchers have developped two strategies: either they "invent" (in some way) the missing component, or they use some higher order differential information. It is this last approach that we explore for the case of rigid curves.

We have particularly taken care to use only the information available generically from the image sequences. The equations relating the kinematic screw associated to the 3D motion to the image measures have been studied carefully. An algorithm that works with synthetic and real sequences have been implemented. Its bringing to life, required the development of methods to compute the needed derivatives (up to order 2). One of the main conclusions that we can draw from these experiments is that, if the computation of the motion on the basis of a single edge contour is possible, it is absolutely necessary to take into account certain constraints (called visibility constraint) that can be imposed upon the solution to verify the property that the corresponding 3D curve is totally in front of the camera.

Key-words: Motion Analysis, Computer Vision, Rigid Curves, Optical Flow, Motion Field

(Résumé : tsvp)

This report is the english version of my *Thèse de Doctorat* defended on May the 10th, 1995, at *Université de Paris-Sud, Orsay.* Stefan Carlsson and Jean-Michel Morel were external referees, and the members of the thesis commitee were: Joseph Mariani (President), Olivier Faugeras (advisor), Michel Merle and Michel Pierre.

Unité de recherche INRIA Sophia-Antipolis
2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex (France)
Téléphone : (33) 93 65 77 77 – Télécopie : (33) 93 65 77 65

Analyse du mouvement de courbes rigides tridimensionnelles à partir de séquences d'images

Résumé : Cette thèse s'attaque au problème difficile de la détermination du mouvement et de la structure d'une scène à partir de séquences d'images dans le cas particulier où celle-ci est constituée d'une courbe rigide. L'intérêt de ce problème est à la fois théorique et pratique : en effet, même si la situation qui nous occupe peut sembler caricaturale, il importe de bien comprendre ce qui se passe dans ce cas qui constitue, en quelque sorte, la situation générique à laquelle on est confronté. Car calculer le mouvement à partir des contours pose des problèmes. Le premier d'entre eux est le problème dit de l'ouverture : en un point de contour seule la composante normale du champ de mouvement image peut être récupérée. Face à ce problème, les chercheurs ont développé deux stratégies : soit on «invente» (d'une certaine manière) la composante du champ de mouvement manquante, soit on utilise des informations différentielles d'ordre supérieur. C'est cette dernière voie que nous explorons dans le cas des courbes rigides.

Nous nous sommes tout particulièrement attaché à n'utiliser que l'information génériquement disponible à partir des images. Les équations liant le torseur cinématique associé au mouvement 3D aux mesures images sont étudiées en détail. Un algorithme marchant avec des séquences tests synthétiques et réelles a été implémenté. La mise en oeuvre de celui-ci a nécessité la mise au point de méthodes de calcul pour les dérivées (jusqu'à l'ordre 2) qui sont nécessaires. Un des principaux enseignements que l'on a pu tirer de ces expériences est que si le calcul du mouvement sur la base de l'observation d'un unique contour est possible, il est cependant indispensable de prendre en compte certaines contraintes (dites contraintes de visibilités) qui imposent à la solution de vérifier la propriété que la courbe 3D correspondante est totalement devant la caméra.

Mots-clé : Vision par ordinateur, Analyse du mouvement, Courbes rigides, Flot optique, Champ de mouvement

Contents

List of Figures

List of Tables

Remerciements

Tout d'abord et avant tout je tiens à remercier **Olivier Faugeras** pour m'avoir accueilli dans son laboratoire et m'avoir proposé ce sujet de recherche. Son enthousiasme pour la vision par ordinateur est extrêmement communicatif. Son soutien tant moral que scientifique m'aura toujours été précieux tout au long de cette thèse.

Je voudrais également exprimer ma gratitude aux membres du jury de thèse : la taille de ce mémoire ne leur a certainement pas facilité la tâche. Ma gratitude va en premier lieu à **Stefan Carlsson** et à **Jean-Michel Morel** pour avoir accepté d'être mes rapporteurs. **Joseph Mariani** m'a fait l'honneur de bien vouloir présider le jury. Un grand merci à **Michel Merle** pour avoir accepté de modifier son emploi du temps afin de pouvoir venir à la soutenance et pour ses encouragements. Merci également à **Michel Pierre** qui a accepté de faire partie de ce jury bien qu'il ne puisse assister à la soutenance. Enfin, je tiens à exprimer ma gratitude envers **Claude Puech** qui devait faire partie de ce jury : je regrette vivement que cela n'ait pu se faire suite au changement de la date de soutenance. Ses encouragements m'ont été droit au cœur.

Toute ma reconnaissance va également à toutes les personnes que j'ai pu cotoyer au sein du projet ROBOTVIS. Leur amitié, leur aide et leurs bons conseils tant scientifiques que personnels m'ont énormément soutenu tout au long de cette thèse. Cette expérience a également été la source de nombreuses discussions extrêmement

enrichissantes. A cet égard, je tiens tout particulièrement à remercier **Nour** et **Éphie**, **Thierry** (l'homme aux pieds nus), **Zhengyou**, **Jean-Luc**, **Hervé**, **Nassir**, **Régis** et **Pascale**, **Tuan**, **Michel**, **Luc**, **Cyril**, **Stéphane**, **Frédéric**, **Gaby**, **Bernard**, **Thierry** (l'homme au détecteur de coins), **Reyes**, **Béné**, **Brent**, **Stefan**, **Charlie** et **Philippa** et, bien sûr, **Marie-Line**.

Je remercie **André Galligo**, **Bernard Mourrain** et **Jean-Pierre Merlet** pour leurs conseils en matière de résolution de systèmes polynômiaux.

Un grand merci tout particulièrement à **Peter**, **Oanh** et **Kaija** pour leur amitié qui m'est précieuse, pour leur soutien constant au cours de cette thèse et pour leur aide durant la rédaction. Ma gratitude va également à **Charlie** qui a grandement contribué à l'amélioration du style de la version anglaise de ce manuscript.

Je suis également redevable aux différents services de l'INRIA auxquels j'ai eu à faire et tout particulièrement « aux filles de la doc », aux personnels du SEMIR, de la cafeteria, du bureau des missions et du bureau du personnel. Leur gentillesse et leur promptitude à régler pour moi certaines situations constituent certainement un des atouts majeurs de l'INRIA.

La mention spéciale du jury est décernée à tous ceux qui m'ont fait l'honneur de m'accorder leur amitié. Les bouffes, sorties, promenades, discussions et autres distractions ont largement contribué à me donner la sérénité nécessaire à la réalisation de ce travail. Merci à vous tous **Vittoria** (viva Pisa), **François** (du bon coté de l'étang de Berre), **Nassir**, **Mahzad**, **Farzhad** et à travers vous toute *l'iranian connection*, **Sylvie**, **Stéphane** (M^r TETRIS), **Xuân-Nam**, **Stéphane** (le breton), **Marie-Claude** (la corse), **Henri** (le sarde), **Monique** et **Olivier**, **Luc** (M^r bowling), **Boni**, **Katherine**, **Jean-Michel**, **Nathan**, **Éric**, **Nathalie**, **Bruno**, **Tilo**, **Suzanne**, **Nicolas**, **Hany**, **Laurence**, **Stephane** (le suisse), **Soraya**, **Roger**, **Diane**, **Olivier**, merci pour tous les bons moments que nous avons partagés. J'espère qu'il y en aura beaucoup d'autres.

Finalement, je tiens à remercier toute ma famille et plus particulièrement mes parents et mon frère pour leur patience et tous leurs encouragements. Tout ce que je suis, je le leur dois...

Chapter 1

Introduction

The goal of three-dimensional computer vision is to build a vision system based on passive sensors. Although it is currently unreachable, the ultimate achievement would be the design of a system that would mimic our (human) visual system: such a tool would be of great interest in the fields relying heavily on human vision such as surveillance, quality control, or robotics. Because of this, and even if not stated explicitly, most of work in the computer vision has relied heavily on some assumption arising from our experience about ours visual system: the results that we show are most often interpreted as images (i.e. re-interpreted in a non-symbolic manner by our visual system), and the performance that we would like to obtain are those which we imagine are readily achievable in our everyday lives.

On the contrary, and as far as our knowledge allows such a statement, the treatment of visual information in the brain is somewhat different from the methods of computer vision. Nevertheless, the input data are similar and the dream of every researcher in computer vision is to obtain a computerized system that would provide output data close to the supposed output of our visual system (even if it is difficult to know exactly what this output is, we do however know quite well what we are able to achieve on its basis: actually, evaluating the quality of an algorithm is maybe

one of the most difficult problem in computer vision as, practically, it can be gauged only on the basis of the subsequent treatments).

From this human centered point of view, and because the study of the biological visual systems is so difficult, one of the most interesting kinds of information that could be obtained from our insight of the human visual system relies on the following assertion:

Biological systems have the tendency to keep only those of the capabilities that are regularly trained.

Although some care has to be taken with such a statement, should we believe in it, then every task that our visual system seems to be able to achieve must be studied within the framework of computer vision. Thus, if possible, tools providing a similar capability should be designed.

Let us now examine the problem that is the topic of the work presented here:

The analysis of the motion of a 3D rigid curve from a monucular sequence of images.

Whoever has tried to close one eye for a while, has experienced the human capability of infering motion and structure from a monocular vision of its environment. Pursuing the experiment further, one can discover how it is possible to do such a job while observing a single smooth curve even when it is planar (which, in computer vision, introduces some mathematical difficulties). This tends to show that some monocular treatment of the visual information is made by the brain and that the particular task of determining the motion of a single piece of curve might play an interesting role in our visual system.

1.1 Motivations

Actually, the justification given in the previous paragraph is very philosophic. There are also some more pragmatic reasons that make the study of the analysis of the motion of a monocular sequence of 3D rigid curves particularly interesting within the framework of computer vision:

- The main idea is to develop a generalization of the works on motion estimation based on points and lines (see for example [LHP80, WU85, WKPS86, Sub88, FDN89, WHA92]). However, with respect to the problem of motion estimation, a single point or a single line is a very "poor" source of information. Combining many of these data is somewhat dangerous as, in general nothing asserts that the underlying 3D motion corresponding to the different primitives is the same: the different primitives may belong to different objects each of which has its own different 3D motion. Moreover, point based approaches are penalized by the fact that detecting points from images is a difficult task that is prone to errors. More structured primitives are easier to recover and to track. All these reasons make the curve case attractive as a complex enough curve is both a richer source of information (i.e. the full 3D motion can be computed from its observation in a monocular sequence) and an easy-to-detect feature (since it is still a quite structured primitive).

- Similarly, a great deal of the work on 3D motion has been done in the case of discrete motions (especially for the straight line case). In many situations, continuous theories are also interesting to develop as they provide the ground for the treatment of images taken in a rapid succession [BBM87]. Making the assumption of continuous motion simplifies the problem of tracking a feature a lot as it moves little from one image to an another. Another argument for working with this kind of image sequence is that it allows the unification of the concepts of time and space in a single spatio-temporal framework.

- One interesting consequence of the estimation of motion is that, once done, it is in theory possible to recover the 3D structure of the observed curve. This would allow the development of a device establishing a 3D structure for which the hardware would be minimal.

- Finally, towards the periphery of the work presented here, and more specifically on the basis of the results obtained in chapter 4, one might wonder what kind of low-level spatio-temporal information is used in the biological visual systems. This is all the more interesting as some works in neurophysiology seem to suggest that these systems are measuring quantities that cannot be recovered mathematically!

1.2 Organization of the Discussion

The discussion is organized into three main parts: a preliminary part introducing the fundamental notions that will be used in the remaining of the discourse, a second part describing the method used to compute some particular quantities needed in the third part which itself studies the problem of the recovery of the motion and the structure of a 3D rigid curve in depth.

In a final part, as a conclusion a brief summary of the contributions of this work is provided. Notice also that, in order to ease the reading of the text, such a conclusion has also been provided for each chapter (except for the first two).

1.2.1 The Preliminary Part

The first part is subdivided in three chapters:

Chapter 2: provides for all the useful mathematical background necessary for the subsequent chapters (essentially chapters 3, 6 and 7). It is designed as a brief reminder of some classical results of geometry (more specifically projective and differential geometry) and of the algebra of polynomial systems. However, some more advanced topics in these fields and also some numerical algorithms are introduced. The goal of the chapter is *not*, however, a complete presentation of these tools and the interested reader is referred to the literature.

Chapter 3: introduces the basic notion and models used in computer vision which are needed for understanding this work. Moreover, this chapter emphasizes some common problems that may arise trying to recover 3D properties from images. Futhermore, it explains some somewhat "arbitrary" choices that were made in the presented work.

Chapter 4: studies in depth the information which is available from a sequence of images for understanding the motion of a 3D non-elastic curve. Actually, this chapter would have found a better place in the third part of the thesis, but it is presented here as it introduces the quantities whose estimation is the topic of the second part. This chapter also proposes a set of stimuli that may be used to clarify the question about what are measuring biological systems. The development of these stimuli is something difficult, so should still be considered as "experimental".

Apart from chapter 4, which contains some important definitions, most of the material of this part covers standard or advanced results in mathematics and computer vision which can be skipped for a first reading. They are necessary, however, for the understanding of some details of the overall presentation.

1.2.2 The Derivative Computation Part

The second and third parts are largely independent: indeed, the estimates obtained in chapter 6 are used in chapter 9, but the means by which these values have been obtained is not relevant for the discussion of the last part. The second part is organized into two chapters:

Chapter 5: presents the problem we want to solve, and gives the general outline of the algorithm defined to solve it. Then, a complete methodology for the validation of such an algorithm is presented along with some of the tools needed to use it effectively. This chapter constitutes the theoretical layer on which chapter 6 is built.

Chapter 6: displays the results obtained with the defined method. Several variants are considered (especially for the case of curvatures) and some critical design issues are discussed. The main goal of this chapter is to prove that it is possible to compute the second order derivatives accurately. These quantities will be used by some of the algorithms presented in the third part. The final sections and the conclusion summarize the interesting remarks that can be made from the results which have been obtained, and propose some extensions that would eventually allow a further improvement of the quality of the results.

In addition, a brief introduction describes the goal of the part and gives a brief summary about derivative computation method and about the use of derivatives in computer vision.

1.2.3 The Motion and Structure for Rigid Curves Part

This last part is the main goal of the work presented. As an introduction, a brief history of the problem of motion and structure (mainly centered on the case of general 3D motions) is presented. The rest of the part is constituted of four chapters:

Chapter 7: studies in depth the general case of a rigid 3D curve in motion. The equations relating the 3D motion to the image parameters are established, and the inter-relations between these different equations are described. The properties of the points of the curves for which these equations degenerate are described. Interesting remarks about the properties of the 3D reconstruction at these points are made.

Chapter 8: gives a detailed study of many different cases which are simpler than the general one. An attempt is made to characterize the complexity of these different problems in terms of the number of zeros of the underlying polynomial systems. These particular cases are divided in two categories: particular motions and particular geometries (special curves). In the cases of planar curves and of ellipses, the importance of the second order information is stressed.

Chapter 9: presents different algorithms for the solution of the motion and structure problems. Attention is mainly focussed on the last one as it is more general. The importance of taking into account of the fact that the points observed on the curve must be visible, is demonstrated in terms of the convergence rate of the algorithm with perfect data. Results are shown for synthetic and real image sequences.

Chapter 10: introduces as a conclusion two possible extensions of the work presented in the part. The first of these addresses the problem of the estimation of motion in the uncalibrated case, whereas the second one deals with the problem of disambiguating stereo matches, by using motion of rigid curves and a stereo setup.

Part I

Preliminaries

Chapter 2

A Few Mathematics

The goal of this chapter is to remind the reader of a few mathematics directly or indirectly needed for reading the next chapters. On the way, some remarks about the particular interest of some specific mathematical objects or techniques in the computer vision framework are made. Most of the time, details are not given here and the interested reader is referred to more consistent mathematical work for a complete view of the topic. The presentation adopted is to define as few things as possible and to give a feeling about "how it works" by making analogies with well known mathematics or with common understanding. However, the section about differential geometry goes into some complex details because they will be heavily used in chapter 4.

2.1 Invariants

Klein in his Erlangen program (1872), revolutionized the study of geometry. He showed how geometry can be seen as the study of the properties of objects (whatever they are) that are invariant under the transformation induced by a group. From this point of view, the "standard" geometries can be classified into a hierarchy characterized by groups that are more and more general. Generalizing the Euclidean

group to, successively, the affine and the projective groups, we obtain geometries that become more and more general. Any projective property is also an affine one. And, in turn, any affine property is also an Euclidean one. Thus, to prove some basic properties, one has the choice of the proper geometry to use. If distances or angles are involved then surely Euclidean geometry is needed, if however only ratio of distances measured along parallel lines or parallels are appearing then the affine geometry should be used. If, finally, only incidence properties appear in the looked for result, then projective geometry is the proper tool. Adapting the level of the geometry to the property which is studied is often very interesting since it allows to forget about the irrelevant details.

This hierarchy of geometries is indeed very important in computer vision. Recovering the properties of the 3D world can be made at many different levels:

- 3D Euclidean properties of the world are often interesting for obtaining quantitative measures required for executing sensori-motor tasks for example. This is the most common level of work in computer vision.
- Affine properties are very interesting to consider in many applications for which only relative distances measured along parallel lines are sufficient. For example, having a robot to follow the center of a corridor needs only an affine reconstruction of it (at least if the corridor is wide enough). This kind of situation also appears when the internal parameters of a camera are not known: this can be considered as inducing an affine transform onto the 3D world.
- Finally, projective properties are sufficient for some recognition procedures or for recovering properties that depend only on incidence. One simple example is that it is possible quite easily (at least theoretically) to decide from two views if a set of points are coplanar or not [Gro94].

Two works characteristic of this level of hierarchies are described in [Fau92, LV94]: the first one shows how it is possible to reconstruct a 3D scene from two uncalibrated images up to an affine or projective transformation. Euclidean reconstructions can also be made but at the cost of adding more information (more images [MF92, Har94] or images with known 3D properties [Tsa87, MBB94]). The second work clarifies this geometric hierarchy in the computer vision framework.

Remark 2.1 *There is one more important level not described in this hierarchy and that might be quite important for computer vision: the topological level. In this*

level, only the neighborhood properties are of importance. Although there is not much work on purely topological aspects, it is however present in a lot of algorithms in computer vision like recognition, curve matching for stereo purposes or low-level image processing.

Notice that the description of geometry made by Klein also applies to less "natural" situations such as differential geometry which is the study of smooth objects under the action of the group of the different parameterizations or to algebraic geometry.

Computing the basic invariants for a given geometric situation is not always an easy task. Either if, in some cases, this task is quite easy (for example the coordinates of a point in a canonical basis formed by some other points are invariants), in some others it requires sophisticated mathematical tools [Gug77, VMPO92, Fau94]. Appendix E shows an example of a non trivial computation using Maple.

In this text, we mainly use invariance in a trivial way to find some measures that are independent of the position of the camera in space (this is used to estimate the quality of our algorithms on real images). We also characterized the invariants quantities arising from situations when a rigid motion is seen deformed by a constant affine transformation. There is, however, a new trend about using invariance in computer vision that makes heavy use of invariants: this tendency actually aims at expressing quantities that are viewpoint invariant and, as such, can be computed in (almost) any image as well as in the 3D world [MZ92, MZF94].

2.2 Projective Geometry

This section introduces the strict minimum of projective geometry that is needed to understand some parts of the following chapters. The purpose here is to introduce in some tutorial way the tools that will be needed. For a complete treatment of projective geometry the reader is referred to [SK52] for example.

2.2.1 Euclidean and Projective Coordinates of Points of $\mathcal{P}^2$

What makes projective geometry so appealing for computer vision is that it is perfectly well adapted to the description of the features in the images under the standard

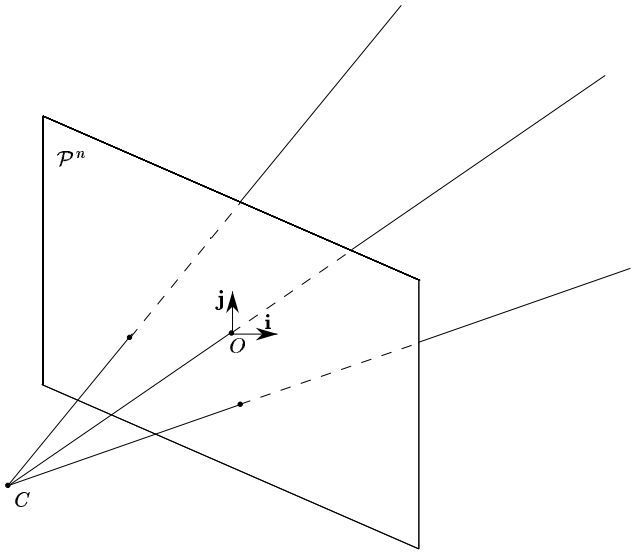

Figure 2.1: An interpretation of the projective space $\mathcal{P}^n$ as the line directions of an $n+1$ dimensional space. This figure represents the case $n = 2$.

pinhole model (see next chapter). Actually, one can interpret the projective space of dimension n (denoted by $\mathcal{P}^n$ hereafter) as the space of the line directions of an Euclidean (or affine) space of dimension $n+1$. Figure 2.1 depicts this situation for the projective plane ($n = 2$). Each line direction is represented by its intersection with the projective plane $\mathcal{P}^n$ and reciprocally each point of $\mathcal{P}^n$ represents a line direction in the $n+1$-dimensional space. We now briefly describe a few basic tools allowing manipulation of projective quantities.

A projective basis of $\mathcal{P}^n$ is constituted by $n+2$ points no n of which being coplanar. The coordinates of a point in such a basis constitues an $n+1$ vector defined up to a non null scale factor: one of its components, at least, is non zero. This coordinates can be viewed as those of a line direction (a vector) into the $n+1$-dimensional space associated to $\mathcal{P}^n$. A transformation between projective spaces is represented by a non-null square matrix of dimension $n+1$ defined up to a scale factor. The transformation is applied by just applying the linear transform represented by the matrix to the projective coordinates of a point. The basic invariant of projective spaces is the cross-ratio which is defined for the configurations of 4 aligned points.

Suppose now that we have an Euclidean (or affine) basis of the 2D space P represented by the frame $(O, \mathbf{i}, \mathbf{j})$ and that we want to embed this Euclidean space into a projective one. The easiest way to achieve such a goal is given by figure 2.1. In this figure, a new point C not in the plane P has been added. This point can be chosen arbitrarily as long as it stays away from the plane. Let us chose it on the normal to P at point O, the distance of C to the plane being 1 (changing the position of C is just like making a change of projective basis, so this simple choice is as good as an another one). Thus, the *projective or homogeneous coordinates* of a point $\mathbf{m}$ of Euclidean coordinates (x, y) is given by the vector $(x, y, 1)$. Of course, $(\lambda x, \lambda y, \lambda)$ is also a valid projective representation of $\mathbf{m}$ as soon as $\lambda \neq 0$. Reversing the same reasoning, we find that the Euclidean coordinates corresponding to the projective representation (x, y, z) of $\mathbf{m}$ in our basis are given by $(x/z, y/z)$. From this it is clear that projective points for which $z = 0$ are not Euclidean points: if we refer to figure 2.1, the line directions corresponding to these points are those that are parallel to the intersecting plane. Thus, they can be interpreted in two different ways: either as vectors of the vector space associated to P (thus, in our basis the projective coordinates of a vector $\mathbf{v} = [a, b]$ are $(a, b, 0)$) or as points that lie in some extension of this plane P that is located at *infinity*. Indeed, these points are called points at infinity and $\mathcal{P}^2$ is the union of P and of the set of these points.

2.2.2 Euclidean and Projective Lines of $\mathcal{P}^2$

Having understood what are the representations for points and vectors, the next object of interest is the line. Following our Euclidean analogy, a Euclidean line considered as a set of points is represented as $\mathbf{m} + \mu\mathbf{v}$. Following the rules defined in the previous section, we find the projective coordinates of the points of the line as

$$\begin{bmatrix} \mathbf{m} \\ 1 \end{bmatrix} + \mu \begin{bmatrix} \mathbf{v} \\ 0 \end{bmatrix} .$$

If now we introduce the projective coordinates $\mathbf{D}$ of the Euclidean normal vector to this line, we notice that for any point on the line which projective coordinates are represented by the vector $\mathbf{x}$, we have:

$$\mathbf{D} \cdot \mathbf{x} = 0 , \tag{2.1}$$

where $\cdot$ represent the standard dot product. $\mathbf{D}$ is said to be the projective representation of the line. Similarly, having the projective coordinates $\mathbf{x}_1$ and $\mathbf{x}_2$ of two points, the projective representation of the line joining these two points is

$$\mathbf{x}_1 \wedge \mathbf{x}_2 \ ,$$

where $\wedge$ represents the standard cross product. Notice that points and lines have the same projective representation in terms of homogeneous coordinates and that equation (2.1) involves the line and the point in a symmetric fashion. This is the so-called *duality* of points and lines in the projective plane. This duality principle just states that manipulating the set of the projective lines is equivalent to manipulating the points of another projective space (the dual space). Thus equation (2.1) can also be seen as the set of all the lines passing through the point $\mathbf{x}$. Similarly, the point intersection of two lines $\mathbf{D}_1$ and $\mathbf{D}_2$ is $\mathbf{D}_1 \wedge \mathbf{D}_2$!

2.2.3 General Projective Spaces

A lot more can be said on the topic of projective geometry. Basically, replacing the word line by hyperplane, all that has been said in the previous two section has a straightforward generalization for the general projective space $\mathcal{P}^n$. Similarly, most of the properties of the projective coordinates can be generalized to any projective basis. Notice, however, that the way to go from Euclidean coordinates to projective ones described in section 2.2.1 is supposing a particular projective basis that can be easily obtained from the Euclidean one. We do not go into deeper details since what has been introduced up to now is sufficient for the use we make of projective geometry in this text. For more details about the tools of projective geometry that can be used in computer vision, the reader is referred to chapter 2 of [Fau93] and to [Kan91].

2.3 Polynomial Systems

Polynomial systems are appearing in almost all the problems of computer vision: this is because the camera model that is used is linear (and thus algebraic) and because most of the characteristics of the 3D world that people want to recover can be modeled quite easily by algebraic conditions. There is however a big difference

between the standard mathematical problem of solving a system of polynomials and the ones that appear in computer vision: on one side, since the measurements are corrupted with noise the polynomials that are manipulated are not exact which normally perturbs the solutions (and this effect can be extremely important). On the other hand, the polynomial systems arising in computer vision are very often overdetermined but they are known to have solutions (contrarily to the mathematical situation into which overdeterminated systems are usually not considered since they do not have, in general, any solution). We are thus facing a non generic problem that is not much studied by mathematicians. The goal of this section is to give a brief overview of the standard methods used to analyze and solve polynomial systems. Unless stated otherwise, the polynomial systems considered here have as many equations as unknowns.

Basically, two big problems are appearing with these systems:

- Counting the number of solutions is an important task since it characterizes the eventual ambiguity of a problem (see for example the work of Faugeras and Maybank [FM90, May90a]) or the troubles that might be encountered while trying to solve such a system.
- Obviously, the other problem is of course to obtain the solutions. The presence of noise does not facilitate this task. On the other hand, the overdeterminacy of the system may help to cope with this problem. Another benefit that might come from this characteristic is the hope to have more efficient solving methods than the general ones.

One fundamental problem that comes in addition to the previous ones is that computer vision usually looks for real solutions only whereas standard mathematics generally consider the situation in the complex field since it is simpler. Real algebraic geometry that studies these real properties is very difficult and is thus not touched upon here. However, one useful result that it states is that, on average, the number of real roots of a polynomial system is the square root of the number of its complex roots [SS93]. This, at least, can give a hint about how to derive a result for the real situation from one found for the complex one.

2.3.1 Symbolic Methods

We give here a brief description of a few symbolic tools that can be used to find some basic information about a polynomial system or that transform it into some

canonical form with which it is much easier to find all the complex solutions. These methods are called "symbolic" because they are essentially manipulations on the form of the equations.

Elimination and the Bezout Formula

Elimination techniques suppress one or more unknowns between the polynomials of the system. The main idea is to reduce the number of unknowns by expressing some of these as (eventually implicit) functions of the others. This is equivalent to project the original solution set onto a subspace of the original space into which it "lives" and to keep some information (the functions) allowing the recovery of the full solution from its projection. This operation decreases the complexity of a system since it reduces simultaneously the number of equations and the number of unknowns. However, in general, this means that the degree of the polynomials of the system is increased since the number of solutions is preserved. Actually, this is stated precisely by the Bezout theorem.

Theorem 2.1 (Bezout) *The number of complex solutions of a generic polynomial system is equal to the total degree of the system.*

The total degree is the product of all the degrees of the polynomials of the system. Notice that this theorem is exact only in the projective situation where all the polynomials are homogeneous (all the monomials have exactly the same degree. This can always be achieved by adding one more unknown). This has the main drawback of introducing many spurious solutions at infinity. Some results shown in the work presented here will show how big this number can be. The Bezout theorem can be refined in the multihomogeneous situation to not take into account some of these solutions at infinity [WMS90, Wam92], but a better solution is described below.

The resultant is the resulting polynomial of an elimination. Many techniques are known to compute the resultant. Most of them involve the computation of the determinant of a matrix. The particular form, size and complexity of the matrix differs from one method to an another. Each kind of matrix (Bezout, Sylvester, Dixon, Macaulay) leads to a method having its advantages and its drawbacks [Dix08, Cha90, GCL93]. There is however a new trend in symbolic computation that aims at efficient implementations of multi-polynomial resultant specially for

sparse polynomials [MC93, Can93]: the basic idea behind this technique is to take advantage of the null coefficients that appear in the polynomial to find a smaller upper bound on the number of solutions. This also leads to methods for solving polynomial systems.

Gröbner Bases

Gröbner basis techniques attempt to solve the problem of representing the ideal associated to the polynomial system in some canonical form[1]. In some way, this can be seen as a generalization of the elimination technique or as the generalization of the standard Euclidean algorithm to multivariate polynomials.

Finding a Gröbner basis for a polynomial system is not a trivial task and a lot of research efforts are made in order to improve the current methods and to speed them up. The most famous algorithm is the Buchberger's algorithm whose complexity is twice exponential in the number of unknowns. This makes it hard to apply to problems where there are more than 6 or 8 variables (a variable is either an unknown or a parameter involved in the polynomials). It is however a useful technique to apply some simplification rules over a formula (see Maple's simplify function with side relations for example). A good implementation of Gröbner bases is provided by Macaulay [SSB89]. The interested reader is referred to [GCL93, CLO92] for a detailed presentation of Gröbner bases.

Newton Polytopes and Bernstein Theorem

As noted previously, there is a big gap between the Bezout bound and the actual number of solutions of a polynomial system. For example, Canny cites the case of a kinematic problem for which the Bezout bound is 46656 whereas it is known that there are not much than 16 solutions. There is thus a lot of improvement to be made. A new technique appeared recently [Ber75] that counts only the finite zeros for which none of the components is zero (Notice that this restriction is easy to

[1]The ideal associated to a polynomial system $\mathcal{S}$ is the set of all the polynomials that can be written as algebraic consequences $\mathcal{S}$. It is, trivially, the object of interest when it comes to work algebraically with the zeros of $\mathcal{S}$. A canonical form is a specific polynomial system characterized by some arbitrary rules that make it essentially unique. This allows to take into account the fact that some polynomials of the system might very well be algebraic combinations of the others. Canonical forms are specially useful to compare formulas to say whether they are equivalent.

work with since it is sufficient to make a translation on the space of the unknown variables).

The basic object manipulated to obtain this bound is the *Newton polytope*.

Definition 2.1 *The Newton polytope of a polynomial P with n unknowns is the convex hull Q (in R^n) of all the monomials of P represented in the space Z^n. The integer coordinates of this representation are the exponents along each of the unknown variables.*

An example of the Newton polytope for a polynomial equation is given in figure 2.2. Notice that only the vertices of the convex hull Q are significant. Also the actual values of the coefficients of the monomial are not taken into account. To go further, we need to define the *Minkowski sum* and the *mixed volume* of convex polytopes.

Definition 2.2 *The Minkowski sum of convex polytopes of two convex polytopes A and B in R^n is the set $A + B = \{a + b | a \in A, b \in B\}$.*

Definition 2.3 *Given a set of n convex polytopes $Q_1, \ldots, Q_n \subset R^n$, there is a unique real-valued function $MV(Q_1, \ldots, Q_n)$ called the mixed volume of $Q_1, \ldots, Q_n$ which is multilinear with respect to the Minkowski sum. Note that $MV(Q, \ldots, Q) = n!Vol(Q)$ where $Vol(Q)$ is the usual Euclidean n-dimensional volume of Q.*

Finally, it is possible to state the *Bernstein theorem*.

Theorem 2.2 (Bernstein) *Let $f_1, \ldots, f_n$ a system of n polynomials in n unknowns. The Newton polytope associated to f_i is denoted by Q_i. The number of the complex zeros with no zero component of the system is either infinite or does not exceed $MV(Q_1, \ldots, Q_n)$. For almost all specialization of the coefficients of the polynomials, the number of solutions is exactly $MV(Q_1, \ldots, Q_n)$ (in the following we adopt the convention $MV(f_1, \ldots, f_n) = MV(Q_1, \ldots, Q_n)$).*

This bound avoids counting the zeros at infinity. For example, applied to the problem mentioned at the beginning of this section it gives a bound of 2304 (or 384 if applied carefully) which is a great improvement over the Bezout bound. Why is

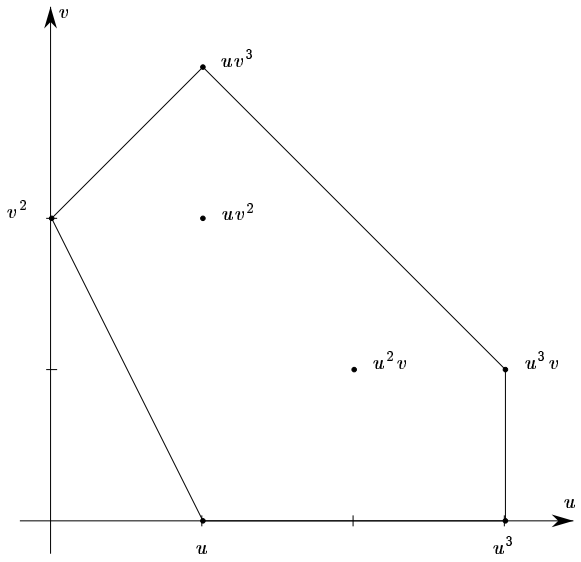

Figure 2.2: The Newton polytope associated to the polynomial $u + 2v^2 - uv^2 + uv^3 + 3u^2v - u^3 + 2u^3v$ (the actual values of the coefficients are not significant as soon as this value remains different from 0. The points corresponding to the monomials are represented by the dots and the Newton polytope is shown in plain lines.

this bound not equal to 16, the best known bound? Basically, this arises from the fact that there might be relations between the coefficients that are not taken into account by the method. Consider for example the two following systems:

$$f_1 = 1 + x + y + xy, \quad f_2 = 1 + 2x + 2y + xy \ ,$$
$$g_1 = 1 + 2u + u^2 - v^2, \quad g_2 = 1 + 4u + u^2 - v^2 \ .$$

The g_i polynomials are obtained from the f_i ones by the change of variables $x = u + v, y = u - v$ but we have $VM(f_1, f_2) = 2 \neq VM(g_1, g_2) = 4$. This illustrates the difficulty of this method. The obtained bound is however always much better than the Bezout's one. In this work, we have used the work of Emiris and Canny described in [EC93]. A special program to compute volumes of convex polytopes was used.

2.3.2 Numerical Methods

It is not always obvious to clearly differentiate the symbolic methods from the numerical ones since more and more practical methods for solving polynomial systems involve both symbolic and numerical aspects. Numerical methods are considered here as those that directly attempt to approximate some of the roots of the polynomial system.

Continuations

Continuations or homotopy method is a quite old method that uses the continuity of the zeros of a system of equations when its coefficients are varied smoothly to find the solutions of a system (the goal system) starting with those (or a subset of those) of another system of equations (the start system). This start system for which the solutions are known must have the same structure as the goal system. This technique has been adapted recently more specifically to polynomial systems yielding a practical method for finding all the solutions of such a system [WMS90].

Basically, this technique uses one of the bounds on the number of solutions described above (Bezout number or Bernstein bound): it computes an initial system $F(\mathbf{x})$ for which all the complex solutions are known, isolated and that has the same

structure as the goal system $G(\mathbf{x})$. What exactly is this structure depends on the used bound: for Bezout's bound the degree of each equation is the structural parameter to preserve whereas for the Bernstein bound it is the vertices of the Newton polytope that need to be preserved. Then a new system

$$H(\mathbf{x}, \lambda) = (1 - \lambda) \exp^{i\theta} F(\mathbf{x}) + \lambda G(\mathbf{x})$$

is formed. When λ varies from 0 to 1 and provided that no singular situation is encountered during the path (these situations can be avoided by choosing θ properly), the zeros of $F(\mathbf{x})$ are mapped one to one onto the zeros of $G(\mathbf{x})$. The basic technique is to start with $\lambda = 0$ and the original zeros of $F(\mathbf{x})$ and then, to make a small step $d\lambda$: the zeros of $H(\mathbf{x}, \lambda)$ are refined into zeros of $H(\mathbf{x}, \lambda + d\lambda)$ using a standard numerical method to find zeros of multivariate systems — usually the Newton method —, the new obtained zeros being the starting zeros for the next $d\lambda$ step. The algorithm finishes when $\lambda = 1$. Of course, things are not that easy since some zeros might diverge or since two paths of zeros might fuse. Moreover, finding the system $F(\mathbf{x})$ might not be an easy task (this is current research specially for homotopy methods used in conjunction of the Bernstein bound [Wam92, VC93]). A good implementation of homotopy methods can be found on Netlib[2] (unfortunately in fortran !) but we used a re-implementation in C of the program described by Morgan in his book [Mor87] coming from the vision group of the University of Illinois.

Exclusion Techniques

There is another kind of numerical method that appeared recently that, contrarily to homotopy methods, deals only with real zeros. This is the so-called exclusion technique. It is just cited here for completeness, since we had no experience with it. Basically, the idea is to look at a finite part of the real domain R^n. For each point of this domain, it is possible to compute a number that characterizes the neighborhood around this point in which it can be asserted that there are no zeros (this number is basically the radius of a ball centered at the point). This number defines a region of R^n which can be excluded from the set of the possible zeros. The method then scans all the domain of interest to a given accuracy to locate (up to

[2] Netlib is a server for mathematical routines. There are many sites all over the world, one of them being netlib.att.com .

that accuracy) all the zeros that lie in this domain. For more details the reader is referred to [DY93, DY91].

Least-Squares Methods

Unfortunately, most, if not all, of the previous methods are unable to cope either with noise on the coefficients or with more equations than unknown variables. To cope with these problems, the numerical analysis community often uses the least-squares technique. This numerical technique replaces the computation of the zeros of a polynomial system (or of any system) by the problem of minimizing an energy function (or error criterion): this criterion is nothing else than the sum of the squares of the original polynomials of the system of equations. There are many methods to perform the minimization (gradient descent, quasi-Newton methods,...) but all of them require an initial value. How to find this value is not considered with this kind of approach (one approach is to first use a continuation method onto a subset of the equations but this might generate much too many initial values).

Notice that the choice of the kind of functional to combine the individual equations is somewhat arbitrary (one can choose for example the sum of the absolute values or the sum of the fourth power of the original equations) and might introduce quite a few "spurious solutions" that are minima of this functional without being zeros of all the original polynomials. A very simple example of that effect is described in figure 2.3. These spurious solutions are not a problem with perfect data (or just data accurate enough) since the value of the criterion at the minimum can be checked: if it is zero then the minimum corresponds to a zero of the polynomial system, if not it is just a spurious minimum. The only problem that remains in this case is thus to be able to find an initial value that leads to (one of) the looked for zeros. With noisy data, the problem is much harder since often zeros are undistinguishable of local minima. Anyhow, least-squares is the only technique that allows to have more equations than unknowns (ignoring the noise and local minima avoidance problems, this certainly reduces the number of solutions of the polynomial system). Moreover, since it performs some kind of "averaging" among the input polynomials, it is also the only method that is able to cope reasonably with the noise problem. One way to avoid the spurious minima problem is either to have a good initialization (in which case the minimization is just a refinement operation) or to have a quite convex problem (i.e. a problem with very few minima: for example, problems with small degrees of freedom or with small degrees are of such a kind).

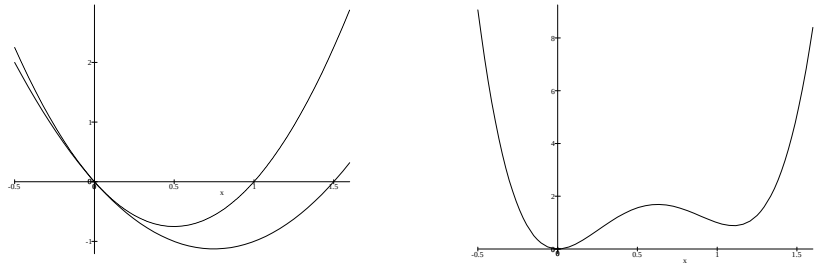

Figure 2.3: A simple example of a "spurious solution" introduced using a minimization scheme: the left plot shows the graphs of the two polynomials $P_1(x) = 3x(x-1)$ and $P_2(x) = 2x(x-3/2)$ and the right plot shows the graph of the sum of their squares. Notice that this graph exhibit a minimum at $x = 1.1$ that is a solution of neither P_1 nor P_2.

2.4 Differential Geometry

This section is a reminder of a few notions of differential geometry in two and three dimensions. Basically, differential geometry is the study of smooth objects and of their invariants. A smooth object is defined[3] by a C^∞ mapping f of an open set U of R^p into an open $V \subset R^n$. f is called a parameterization of V and differential geometry is the study of the invariant properties of V under all the possible parameterizations. Notice that it is possible to add more structure on R^n, the space into which V is embedded. This give rise to more invariants. For example, in what follows, R^n will always be considered as an Euclidean space and this introduces in addition of the tangential properties (which are the basic differential invariants) some more invariants such as length, normals, curvatures, torsion and Frenet frames (which are metric invariants arising from the Euclidean structure). In the remaining of this section, we examine briefly both the purely differential properties and the Euclidean ones of the situations of planar and space curves and surface patches.

[3] This definition has deliberately been made simpler than the usual one. It is, however, perfectly sufficient for our purpose. For a more general definition of differential varieties (the smooth objects) see [Spi79, BG88].

2.4.1 Planar Curves

A planar curve (c) is defined by a mapping f for which $n = 2$ and $p = 1$. This mapping associates to a real u of an open set U of R a point $\mathbf{m}(u)$ of R^2 and when u describes U, $\mathbf{m}(u)$ describes a curve of R^2. Among all the parameterizations of (c), it is possible to distinguish a special family for which $\|\frac{d\mathbf{m}}{ds}\| = 1$. A parameter s that verifies this property is called an arclength. Given such a parameter s, we have the well-known two-dimensional *Frenet formulas*:

$$\frac{d\mathbf{m}}{ds} = \mathbf{t} \ , \quad \frac{d\mathbf{t}}{ds} = \kappa\mathbf{n} \ , \quad \frac{d\mathbf{n}}{ds} = -\kappa\mathbf{t} \ , \tag{2.2}$$

where $\mathbf{t}$ and $\mathbf{n}$ are the tangent and normal unit vectors to (c) and κ is the curvature (the inverse of the radius of curvature) at $\mathbf{m}$ the considered point. $(\mathbf{m}, \mathbf{t}, \mathbf{n})$ is the Frenet frame attached to the curve at $\mathbf{m}$, the sign of κ is chosen in such a way that this frame is oriented in the direct way (by definition this frame is always orthonormal). Table 2.1 gives the formulas for θ (the angle between the normal and a fixed direction of the plane) and κ when the curve (c) is defined either implicitly or in a parameterized way. Another useful formula is $\kappa = \frac{d\theta}{ds}$.

	Parameterized definition	Implicit definition
(c)	$\mathbf{m}(u) = (x(u), y(u))$	$F(\mathbf{m}) = 0$
θ	$tan^{-1}\left(-\frac{x'}{y'}\right)$	$tan^{-1}\left(\frac{F_y}{F_x}\right)$
κ	$\frac{x'y'' - x''y'}{\sqrt{x'^2 + y'^2}^3}$	$\frac{2F_xF_yF_{xy} - F_{x^2}F_y^2 - F_{y^2}F_x^2}{\sqrt{F_x^2 + F_y^2}^3}$

Table 2.1: Formulas giving θ and κ for both implicit and parameterized definition of a curve.

2.4.2 Space Curves

A space curve (C) is defined by a mapping f for which $n = 3$ and $p = 1$. This mapping associates to a real u of an open set U of R a point $\mathbf{M}(u)$ of R^3 and when u describes U, $\mathbf{M}(u)$ describes a curve of R^3. Again it is possible to define the arclength family by considering the parameterizations for which $\|\frac{d\mathbf{M}}{dS}\| = 1$. Given such a parameterization S, we have the well-known three-dimensional *Frenet formulas* for (C):

$$\frac{d\mathbf{M}}{dS} = \mathbf{T} \ , \quad \frac{d\mathbf{T}}{dS} = \kappa\mathbf{N} \ , \quad \frac{d\mathbf{N}}{dS} = -\kappa\mathbf{T} - \rho\mathbf{B} \ , \quad \frac{d\mathbf{B}}{dS} = \rho\mathbf{N} \ , \tag{2.3}$$

where $\mathbf{T}$,$\mathbf{N}$ and $\mathbf{B}$ are respectively the tangent, the normal and the binormal unit vectors to (C) at $\mathbf{M}$ the considered point. κ is the curvature and ρ is the torsion at this point. $(\mathbf{M}, \mathbf{T}, \mathbf{N}, \mathbf{B})$ is the three-dimensional Frenet frame attached to (C) at $\mathbf{M}$. Notice that a change of orientation of (C) (i.e. changing S into $-S$) transforms the Frenet frame $(\mathbf{M}, \mathbf{T}, \mathbf{N}, \mathbf{B})$ into $(\mathbf{M}, -\mathbf{T}, \mathbf{N}, -\mathbf{B})$.

In this case, κ is always a positive quantity, it is the sign of ρ that is chosen in such a way that the Frenet frame is oriented in the direct way. Since a planar curve is also a space curve this seems to be uncompatible with the fact that κ defined in the previous section is a signed quantity. Actually, choosing carefully the orientations of the plane and of the 3D space makes these quantities compatible. For completeness, we give the formulas to compute κ and ρ for parameterized curves:

$$\kappa = \frac{\|\mathbf{M}' \wedge \mathbf{M}''\|}{\|\mathbf{M}'\|^3} \ , \quad \rho = \frac{(\mathbf{M}', \mathbf{M}'', \mathbf{M}''')}{\|\mathbf{M}' \wedge \mathbf{M}''\|^2} \ .$$

2.4.3 3D Surface Patches

A surface patch (Σ) is defined by a mapping f for which $n = 3$ and $p = 2$. This mapping associates to the two reals u and v describing an open set U of R^2 a point $\mathbf{P}(u, v) = \mathbf{M}(u, v)$ of R^3 and when (u, v) describes U, $\mathbf{P}(u, v)$ describes a surface patch of R^3. The normal $\mathbf{N}_P$ to (Σ) at point $\mathbf{P}$ is the unit vector defined as:

$$\mathbf{N}_P = \frac{\mathbf{P}_u \wedge \mathbf{P}_v}{\|\mathbf{P}_u \wedge \mathbf{P}_v\|} \ , \tag{2.4}$$

where $\mathbf{P}_u = \frac{\partial \mathbf{P}}{\partial u}$ and $\mathbf{P}_v = \frac{\partial \mathbf{P}}{\partial v}$.

The Euclidean properties of such a patch are characterized by two quadratic forms called the *the two fundamental forms*, which are defined at every point of (Σ).

The first fundamental form $\mathbf{\Phi}_1$ describes how the Euclidean metric is embedded in the tangent plane $\mathbf{T}_P$ (that is spanned by $\mathbf{P}_u$ and $\mathbf{P}_v$) to (Σ) at a given point $\mathbf{P}$. Basically, it defines the squared length of a vector of $\mathbf{T}_P$ given as $\lambda\mathbf{P}_u + \mu\mathbf{P}_v$. The expression of $\mathbf{\Phi}_1$ is thus:

$$\mathbf{\Phi}_1(\lambda\mathbf{P}_u + \mu\mathbf{P}_v) = \lambda^2 E + 2\lambda\mu F + \mu^2 G \ ,$$

where E, F and G are defined as:

$$E = \|\mathbf{P}_u\|^2 \quad F = \mathbf{P}_u \cdot \mathbf{P}_v \quad G = \|\mathbf{P}_v\|^2 \tag{2.5}$$

Notice that $\|\mathbf{P}_u \wedge \mathbf{P}_v\|^2 = EG - F^2$.

The second fundamental form $\mathbf{\Phi}_2$ is related to curvature. To state it more precisely, let us consider a curve drawn onto (Σ) that goes through $\mathbf{P}$. Locally, such a curve is characterized by its tangent at $\mathbf{P}$. This tangent is in the tangent plane at $\mathbf{P}$ and can be defined by the vector $\mathbf{x} = \lambda\mathbf{P}_u + \mu\mathbf{P}_v$. Now the curvature of this curve at $\mathbf{P}$ is made of two components: one that depends on the variation of λ and μ and another one that depends only on the values of these quantities (i.e. it is the same for all the curves drawn onto (Σ) that have the same tangent direction at $\mathbf{P}$). This last quantity is called the *normal curvature* since it is the projection of $\kappa\mathbf{N}$ onto $\mathbf{N}_P$. This normal curvature is defined by the ratio $\frac{\mathbf{\Phi}_2(\mathbf{x})}{\mathbf{\Phi}_1(\mathbf{x})}$, where

$$\mathbf{\Phi}_2(\lambda\mathbf{P}_u + \mu\mathbf{P}_v) = \lambda^2 L + 2\lambda\mu M + \mu^2 N \ ,$$

L, M and N being defined as:

$$L = \frac{\partial^2\mathbf{P}}{\partial u^2}\cdot\mathbf{N}_P \quad M = \frac{\partial^2\mathbf{P}}{\partial u\partial v}\cdot\mathbf{N}_P \quad N = \frac{\partial^2\mathbf{P}}{\partial v^2}\cdot\mathbf{N}_P \tag{2.6}$$

The invariants of $\mathbf{\Phi}_2$ (under all parameterizations) are important to study. These can be found as the invariants of the linear mapping $\psi : \mathbf{T}_P \rightarrow \mathbf{T}_P$ defined by $\mathbf{\Phi}_2(\mathbf{x}) = (\mathbf{x})\cdot\mathbf{x}$. ψ is called the Weingarten mapping associated to $\mathbf{\Phi}_2$.

Principal Curvatures and Principal Directions

The principal curvatures are the eigenvalues of ψ. They are given as the two solutions of the following quadratic polynomial:

$$(EG - F^2)\sigma^2 - (LG + EN - 2FM)\sigma + LN - M^2 = 0 \ .$$

In particular, the half-sum H and the product K of the two principal curvatures have very simple expressions:

$$\begin{aligned} H &= \frac{1}{2}\,\frac{LG + EN - 2FM}{EG - F^2}\,, \\ K &= \frac{LN - M^2}{EG - F^2}\,. \end{aligned}$$

H and K are respectively called the mean and the Gaussian curvatures. All the invariants of $\mathbf{\Phi}_2$ are functions of these.

To each principal curvature, there is an associated principal direction which is given by the eigenvector associated to the eigenvalue. In the coordinate system $(\mathbf{P}_u, \mathbf{P}_v)$, their coordinates (λ, μ) are solutions of the following equation:

$$(FL - EM)\lambda^2 + (GL - EN)\lambda\mu + (GM - FN)\mu^2 = 0\,.$$

Up to scale factor, this yields the following values for λ and μ:

$$\lambda = EN - GL + \varepsilon\sqrt{\Delta}\,, \tag{2.7}$$

$$\mu = 2(FL - EM)\,, \tag{2.8}$$

where $\varepsilon = \pm 1$ and $\Delta = (GL - EN)^2 - 4(FL - EM)(GM - FN)$.

The Bonnet Theorem

Actually, it is not necessary to introduce the quadratic forms of greater order. The Bonnet theorem states precisely how a surface patch is characterized, up to a rigid motion, by its first and second fundamental forms. Before recalling this result let's introduce the Gauss and Mainardi-Codazzi equations that are involved in this theorem.

These equations come from writing conditions of integrability of the derivatives of the vectors $\mathbf{P}_u$, $\mathbf{P}_v$ and $\mathbf{N}_P$ expressed in the basis these three vectors are constituting. They can be written as in [DoC76]:

$$FK = \frac{\partial \Gamma_{12}^1}{\partial u} - \frac{\partial \Gamma_{11}^1}{\partial v} + \Gamma_{12}^2\Gamma_{12}^1 - \Gamma_{11}^2\Gamma_{22}^1 \,, \tag{2.9}$$

$$\frac{\partial L}{\partial v} - \frac{\partial M}{\partial u} = L\Gamma_{12}^1 + M(\Gamma_{12}^2 - \Gamma_{11}^1) - N\Gamma_{11}^2 \,, \tag{2.10}$$

$$\frac{\partial M}{\partial v} - \frac{\partial N}{\partial u} = L\Gamma_{22}^1 + M(\Gamma_{22}^2 - \Gamma_{12}^1) - N\Gamma_{12}^2 \,. \tag{2.11}$$

The first equation is referred as the Gauss equation while the two others are called the Mainardi-Codazzi equations.

The coefficients $\Gamma_{ij}^k, i, j, k = 1, 2$ are called the Christoffel symbols of the second type. They can be easily computed from the coefficients of the first fundamental form and their derivatives. For the details of their computation see [DoC76].

Theorem 2.3 (Bonnet (local form)) *Given six differentiable functions E, F, G, L, M and N defined on an open set V of R^2, satisfying the conditions $E > 0$, $G > 0$, $EG - F^2 > 0$ and the Gauss and Mainardi-Codazzi equations, then for every $(u, v) \in V$ there exists a diffeomorphism from a neighborhood $U \subset V$ of (u, v) in R^3 such that the corresponding regular patch has E, F, G, L, M and N as coefficients of its first and second fundamental forms. Furthermore, if U is connected, for every other such diffeomorphism, the two patches are related by a rigid transformation of R^3.*

The proof can be found in [DoC76], for example.

Lie Derivatives

Let $\mathbf{V}$ be a function defined on (Σ) such that $\mathbf{V}(P)$ is a vector of $\mathbf{T}_P$. $\mathbf{V}$ is called a tangential vector field. It is very convenient to consider the vectors of $\mathbf{T}_P$ as *differential operators* acting on the functions defined on (Σ). For example, $\mathbf{P}_u$ is the partial derivative with respect to u. We represent this action by $\mathbf{P}_u f = \frac{\partial f}{\partial u}$ and it becomes even more clear if we use the notation $\mathbf{P}_u = \frac{\partial}{\partial u}$. Each vector $\mathbf{V}$ of $\mathbf{T}_P$ is a linear combination of $\mathbf{P}_u$ and $\mathbf{P}_v$ which we write:

$$\mathbf{V} = \alpha\frac{\partial}{\partial u} + \beta\frac{\partial}{\partial v} \,.$$

α and β are the coordinates of $\mathbf{V}$ in the basis $(\mathbf{P}_u, \mathbf{P}_v)$ of $\mathbf{T}_P$. From this it is clear that the action of $\mathbf{V}$ upon a function f which we note $L_{\mathbf{V}} f$ is defined by:

$$L_{\mathbf{V}} f = \alpha \frac{\partial f}{\partial u} + \beta \frac{\partial f}{\partial v} \ .$$

$L_{\mathbf{V}} f$ is the Lie derivative of f in the direction $\mathbf{V}$ of $\mathbf{T}_P$. From this definition follows immediately the following relation:

$$L_{\alpha_1 \mathbf{V}_1 + \alpha_2 \mathbf{V}_2} f = \alpha_1 L_{\mathbf{V}_1} f + \alpha_2 L_{\mathbf{V}_2} f \ .$$

Lie Brackets

Given two vector fields $\mathbf{V}$ and $\mathbf{W}$ on (Σ), we can consider the operator $L_{\mathbf{V}} L_{\mathbf{W}}$. It is a linear operator defined for functions f defined on (Σ) but unfortunately it is not a derivation since it does not satisfy Leibniz's rule. On the other hand, it is easy to show that the *commutator* $L_{\mathbf{V}} L_{\mathbf{W}} - L_{\mathbf{W}} L_{\mathbf{V}}$ is a derivation and therefore a vector field which is denoted $[\mathbf{V}, \mathbf{W}]$ and called the *Lie bracket* of $\mathbf{V}$ and $\mathbf{W}$. By definition, we have:

$$L_{[\mathbf{V},\mathbf{W}]} f = L_{\mathbf{V}} L_{\mathbf{W}} f - L_{\mathbf{W}} L_{\mathbf{V}} f \ .$$

This allows us to compute the coordinates of $[\mathbf{V}, \mathbf{W}]$ in the basis $(\mathbf{P}_u, \mathbf{P}_v)$ of $\mathbf{T}_P$, from those of $\mathbf{V}$ and $\mathbf{W}$:

$$[\mathbf{V}, \mathbf{W}]^i = \sum_{j=1}^{2} V^j \frac{\partial W^i}{\partial u^j} - W^j \frac{\partial V^i}{\partial u^j} \quad i = 1, 2 \ , \tag{2.12}$$

where $u^1 = u$, $u^2 = v$ and V^i (respectively. W^i) indicates the component of $\mathbf{V}$ (resp. $\mathbf{W}$) along the i-th basis vector of $\mathbf{T}_P$ ($\mathbf{P}_u$ if $i = 1$, $\mathbf{P}_v$ if $i = 2$).

Chapter 3

Cameras, Images and Edges

The purpose of this chapter is to give a very brief introduction on the basic models and tools used in image manipulation. These include camera modeling, image formation, motion representation and edge detection. Emphasis is made on the tools that will be used later in this manuscript. As such, this chapter is not intended to be a review of all the methods and models that exist on the topics considered. Similarly, it is not an introduction to computer vision. However, to explain some choices that are made in the sequel of this text, some methods that will not be used are briefly introduced and discussed. Background material for computer vision can be found in [Mar82, Fau93].

3.1 About Cameras

Central to almost all work in computer vision is the notion of camera. Basically, a camera is just a projective device that maps the Euclidean 3D world onto a 2D surface. This description encompasses any imaging process that does such a mapping: biological systems as the human eye, or artificial ones like video-cameras. Why do we call these projective devices? Because if one thinks about image formation, one immediately notice — as people doing e.g. optics or ray tracing know perfectly well

— that only the "light" rays going from an object to the "eye" are important. The analogy with figure 2.1 is striking. Consequently, the geometry of the directions of the light rays plays a central role in the acquisition process and this is exactly why projective geometry is useful.

Moreover, it is difficult to restrict this projective model to a simpler one, such as orthogonal projection, since such a model would not be able to cope with all the possible situations. Notice, however, that such models are perfectly adapted to the limit case where the distance from the view point to the scene is big compared to the sizes of the observed objects: then, the observed objects can be considered as being in the plane at infinity and it is well-known that obtaining tridimensional informations about those is extremely difficult and often involves some other techniques (this is usually the case for example for microscopes or telescopes).

3.1.1 The Pinhole Camera Model

The pinhole camera is the simplest model that fully takes into account the projective nature of a camera. In this model, the camera is modelled by a point O called the *optical center* through which all the light rays are passing and by a plane $\mathcal{R}$ called the *retina* that is the surface on which the image is formed[1]. The retina is assumed to be at a distance f (called the *focal length*) of the optical center.

Remark 3.1 *There are more complex models that introduce some non-linear behaviors such as radial distortion or de-centering. These will not be considered here since it can be assumed that working with good enough lenses and far enough from the borders of the images, these effects can be neglected as a first approximation. If it is needed, it will always be possible to correct the distortion prior to any other use of the image data.*

The 2D projection $\mathbf{m}$ of a 3D point $\mathbf{M}$ is the intersection of the light ray $O\mathbf{M}$ with the retina $\mathcal{R}$. Such an operation is called a perspective projection and is summarized in figure 3.1.

A frame (O, X, Y, Z) is attached to the 3D space in the following way:

[1] The actual form of this surface does not really matter so the model uses the simplest possible kind of surface. Some people prefer to work with spherical retinas but as long as only mathematics are involved this is totally equivalent to the model described here.

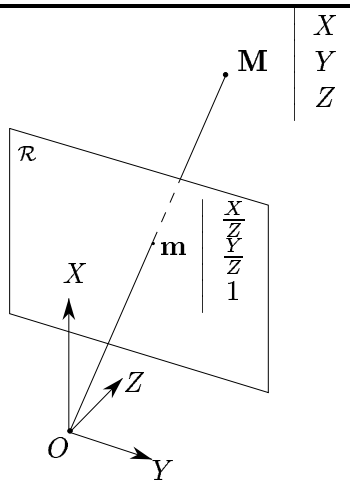

Figure 3.1: The Pinhole Camera Model.

- The retina $\mathcal{R}$ is parallel to the plane (O, X, Y).
- The axis OZ is directed towards the retina $\mathcal{R}$.
- The frame (O, X, Y, Z) is oriented in the direct way.

Remark 3.2

- *Up to a scale factor on the coordinates (X, Y, Z), it is always possible to assume that $f = 1$. This assumption will always be made hereafter.*
- *From the mathematical point of view, the previous constraints define only a one dimensional family of frames, all of them being rotationnaly equivalent. However, physics is not mathematics and this often yields some more constraints. For example, using CCD cameras, it is generally assumed that the axes OX and OY are aligned with the sensing elements of the CCD.*

Since this frame is somehow "naturally" attached to this camera model, all the equations that involve 3D spatial coordinates will be written in this frame.

With the previous notations, the transformation that gives the image point $\mathbf{m}$ of a 3D point $\mathbf{M}$ is characterized by the equation:

$$\mathbf{M} = Z\mathbf{m} \; . \tag{3.1}$$

This equation is fundamental since (as it seems natural) all the constraints that will be written in part III are direct consequences of it. From now on, we use X,Y,Z for the coordinates of the 3D point $\mathbf{M}$. We will also need a coordinate system in $\mathcal{R}$, the situation is, however slightly more complex.

3.1.2 Image Coordinates Versus Normalized Coordinates

When working with images, it is convenient to reference each pixel of the image by its integer coordinates locating it in the matrix of all the pixels. Thus, at least two coordinate systems can be defined onto the retina:

- One that is inherited from the Euclidean structure of the 3D space. It is natural to use this coordinate system as soon as it comes to relate 3D characteristics to their 2D images. The origin of this coordinate system is at the orthogonal projection of the optical center O onto the retina. The axes are taken to be parallel to OX and OY. In the following the coordinates of $\mathbf{m}$ are called *normalized coordinates* and will be referred as x and y.
- One that reflects the array structure of the image. The origin of such coordinate system lies at a corner of the image (usually the top left corner) and the axes are parallel to the row and column directions of the array. The coordinates of $\mathbf{m}$ in this frame are called *image coordinates* and will be referred as u and v.

The relations between these two coordinate systems have been studied a lot in the works treating about *camera calibration* (see below). In the remaining, we adopt the model defined in [Tos87] which is of common use in the computer vision community:

$$x = h_1 u + h_2 v + h_3 \; , \tag{3.2}$$

$$y = h_4 v + h_5 \; . \tag{3.3}$$

The parameters $h_i, i = 1 \ldots 5$ are called the internal parameters of the camera. Equations (3.2) and (3.3) define, in general, an affine transform of the retinal plane.

Consequently, the two coordinate systems have a very different status. *As soon as it is needed to speak of Euclidean properties, the proper coordinate system to work with is the normalized one since these properties are not all preserved by an affine transformation.* In the remaining chapters, unless it is explicitly stated otherwise the normalized coordinate system is always used. Notice also that the projective representation is always used for retinal objects, so those are represented by vectors with 3 components (the third one being 1 for points and 0 for vectors).

Remark 3.3 *The importance of working in the proper coordinate system must not be underestimated. An example of great interest for the matter covered in this work is optical flow: it is well known that along an edge, locally, only the normal flow component can be recovered. This is called the aperture problem (see for example [Hil83a] for a complete description). However, most (if not all) algorithms that compute optical flow directly from images (see for example the implementations of [BFB94][2]) use image coordinates without any camera calibration data. Figure 3.2 shows that the error on the angular direction can be as bad as* 20° *for a typical real camera. Futhermore, notice that, even in a given family of frames (projective, affine or Euclidean), the choice of a particular system of coordinates have also numerical consequences: Hartley obtained recently results that indicate that numerical errors in the computation of the fundamental matrix is correlated to the coordinate system adopted for the image [].*

3.1.3 Calibration

When working with real cameras with extraction of 3D features in mind, it is generally necessary to find the internal parameters that govern the mapping of the 3D Euclidean structure onto the retina. This operation is performed by a process called *calibration* [Tos87, Tsa87]. More generally this operation computes a 3×4 matrix that represent in some arbitrary coordinate system simultaneously the perspective projection and the retinal transformation from normalized to image coordinates. The principle of these method is usually to take an image of a perfectly known 3D pattern and to find the overall transformation from the constraints relating the 2D

[2] This example is given because the code is widely available on the Internet. Note that this might not change most of the quantitative results of the paper since these were obtained using synthetic images for which the horizontal and vertical scale factor might be the same. Only the results obtained from real images might be affected by this problem.

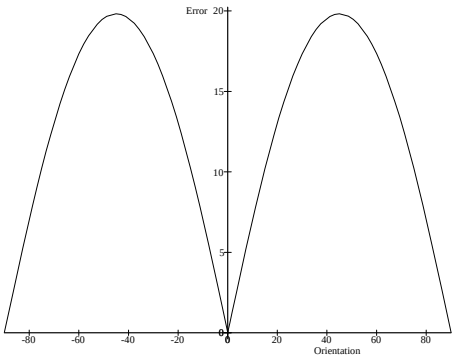

Figure 3.2: The angular error made on the normal as a function of the orientation of the edge. All the angles are given in degrees. The maximal errors are obtained for orientations of $\pm 45°$: this maximal error is around $20°$! The internal parameters used for this figure are those obtained from the calibration of a real camera (Sonny CCD 75CE). Note that this curve is independent of the focal length, it characterizes essentially the geometry of the CCD sensors and its sampling.

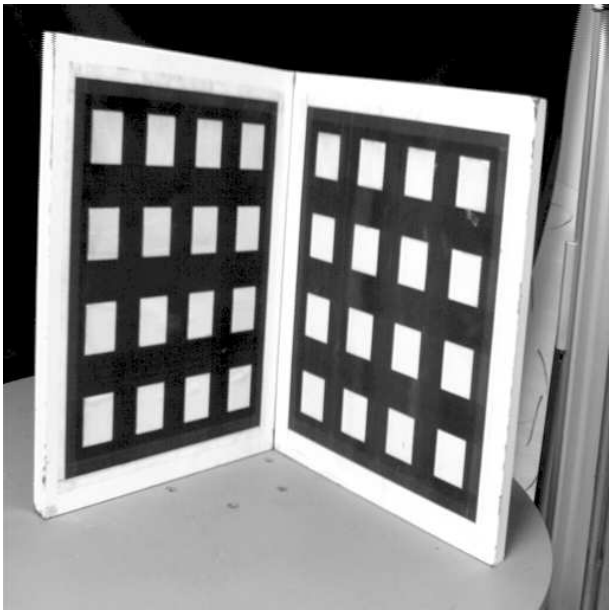

Figure 3.3: The INRIA calibration pattern.

image coordinates to their 3D correspondents. An example of such a pattern is shown in 3.3.

Remark 3.4 *There is, however, a new trend in computer vision that attempts to obtain calibration data from points matches without any necessary a priori 3D knowledge. Although the first results that have been obtained [Luo92, Har94] are very encouraging, such methods are not yet of everyday use, so for the purpose of the work described here, we still work with the pattern based method.*

Of course, since most of the work described here is done in the camera frame, we are not interested in the full 3×4 matrix but only in the internal parameters that can be extracted from it. See chapter 3 of [Fau93], for example, for a complete study of camera modeling and calibration including the formulas that give the internal parameters from the coefficients of the general projection matrix.

3.2 Multiple Camera Representation

A vast majority of methods that compute 3D structure from 2D images combine, to achieve such a goal, information coming from many different view points. To do so, it is necessary to be able to refer to all these information in a global common coordinate system. As we have seen in the previous section, the calibration process is able to compute the transformation that holds between 3D objects coordinates expressed in some arbitrary frame and their images in image coordinates. We present briefly here different methods to represent the relative positioning of the cameras to each other in such situations.

3.2.1 Discrete Situation

In the previous section, we have seen that it is possible to associate a special frame to any camera obeying the pinhole model. This frame along with the projection equation (3.1) totally represent this model. So having multiple cameras is equivalent to having many such frames. This means that the relative positions of the cameras can be represented by the relative positions of the frames and, since these belong to an Euclidean world, the transformation that relate them is the combination of a rotation and a translation. So suppose that we have a global reference frame into which the coordinates of a point are $\mathbf{M}_{ref}$, then using the notation $\mathbf{M}$ for the coordinates of this same point in some camera frame, we have:

$$\mathbf{M} = \mathbf{R}\mathbf{M}_{ref} + \mathbf{T} \, ,$$

where $\mathbf{R}$ is a 3×3 rotation matrix and $\mathbf{T}$ is a 3D vector representing the translation. Obviously, the formula (3.1) must be changed according to this transform in order to be valid in the reference frame.

The previous formulation is perfectly adapted for tasks such as:

- Stereo that combines images taken by 2,3 or more different cameras at a same time instant to recover the 3D structure.
- Pose determination or discrete structure from motion that combine images taken by the same camera at different time instants to obtain 3D information assuming that the scene is static.

This scheme is however not sufficient for dynamic situations into which a changing scene is observed continuously. In such situations, the transformation is a first order quantity (the zero order quantities being the identity for the rotation and null translation).

3.2.2 Continuous Situation

Continuous situations are those for which the camera moves continuously over time. Practically, this means that the time between two frames is very small in comparison with the speed of the camera. This also means that the camera does not move too much between two frames. Thus, for continuous situations, more than the relative position, it is the evolution of the position of the camera that is important. This is usually achieved by representing the velocity of the frame associated to the camera. This is done using a kinematic screw. Since most of the work dealing with motion that is presented here assumes that we are in such a continuous situation (this is sometimes called dynamic vision), we are insisting on this representation and on its properties.

3D motion representation

The field of the 3D velocities associated to the motion of a rigid body is totally described by the expression of a kinematic screw at any point of 3D space (it is assumed that this 3D point is tied to the rigid body). This kinematic screw has two components $(\mathbf{\Omega}, \mathbf{V})$:

- $\mathbf{\Omega}$ is the instantaneous rotational velocity. It is independent of the point at which the screw is expressed.
- $\mathbf{V}$ is the instantaneous translational velocity. It depends on the point at which the screw is expressed.

In the remaining of this text, the kinematic screw $(\mathbf{\Omega}, \mathbf{V})$ is always expressed at the optical center O of the camera.

Two classical results will be needed hereafter:

Proposition 3.1

- *The velocity* $\mathbf{V_M} = \dot{\mathbf{M}}$ *of every point* $\mathbf{M}$ *considered as tied to the rigid body is given by the formula:*

$$\mathbf{V_M} = \dot{\mathbf{M}} = \mathbf{V} + \mathbf{\Omega} \wedge \mathbf{M} \; . \tag{3.4}$$

- *For every vector* $\mathbf{T}$ *of constant norm over time that is tied to the rigid body, we have:*

$$\dot{\mathbf{T}} = \mathbf{\Omega} \wedge \mathbf{T} \; . \tag{3.5}$$

Let us, first, extend this last formula to the case of any non zero 3D vector $\mathbf{W}$. Considering the unit norm vector $\mathbf{T}$ defined by:

$$\mathbf{W} = \|\mathbf{W}\|\mathbf{T} \; ,$$

and deriving it with respect to time, we have:

$$\dot{\mathbf{W}} = \|\dot{\mathbf{W}}\|\mathbf{T} + \|\mathbf{W}\|\dot{\mathbf{T}} \; . \tag{3.6}$$

From equation (3.5), we get $\dot{\mathbf{T}} = \mathbf{\Omega} \wedge \frac{\mathbf{W}}{\|\mathbf{W}\|}$. Furthermore, since $\|\mathbf{W}\| = (\mathbf{W}\cdot\mathbf{W})^{\frac{1}{2}}$, we have:

$$\|\dot{\mathbf{W}}\| = \frac{\dot{\mathbf{W}}\cdot\mathbf{W}}{\|\mathbf{W}\|} \; .$$

Replacing $\dot{\mathbf{T}}$ and $\|\dot{\mathbf{W}}\|$ by their values in equation (3.6) yields:

$$\begin{aligned} \|\mathbf{W}\|^2 \left(\dot{\mathbf{W}} + \mathbf{W} \wedge \mathbf{\Omega}\right) - \left(\dot{\mathbf{W}}\cdot\mathbf{W}\right)\mathbf{W} &= \\ \|\mathbf{W}\|^2\dot{\mathbf{W}} - \left(\dot{\mathbf{W}}\cdot\mathbf{W}\right)\mathbf{W} + \|\mathbf{W}\|^2\mathbf{W} \wedge \mathbf{\Omega} &= \mathbf{0} \; . \end{aligned}$$

Recognizing $\mathbf{W} \wedge \left(\dot{\mathbf{W}} \wedge \mathbf{W}\right)$ in the first term of this last equation, we obtain:

$$\mathbf{W} \wedge \left(\dot{\mathbf{W}} \wedge \mathbf{W} + \|\mathbf{W}\|^2\mathbf{\Omega}\right) = \mathbf{0} \; . \tag{3.7}$$

Taking the cross-product of this last equation with $\mathbf{W}$ yields:

$$\mathbf{W} \wedge \left(\dot{\mathbf{W}} + \mathbf{W} \wedge \mathbf{\Omega} \right) = \mathbf{0} \ ,$$

since $\|\mathbf{W}\| \neq 0$, $\mathbf{W}$ being non zero. This last equation is equivalent to equation (3.7) since the first one implies the second and since developing it and taking its cross-product by $\mathbf{W}$, equation (3.7) is found again. Finally, we have the following proposition:

Proposition 3.2 *For every vector* $\mathbf{W}$ *associated to the rigid body, we have:*

$$\mathbf{W} \wedge \left(\dot{\mathbf{W}} + \mathbf{W} \wedge \mathbf{\Omega} \right) = \mathbf{0} \ . \tag{3.8}$$

Remark 3.5 *All the properties of the velocity field associated to a rigid body come from the invariance of Euclidean distances under rigid motion. The kinematic screw have some more properties. We describe some of those in chapter 8.*

3.3 Image Formation

Although it is a topic that is not often addressed, image formation is very important. Many people just assume that a pixel is a data point with a grey value associated to it. However, physically, such assumptions are only true as a first approximation:

- First the CCD sensitive cells are not points. They are usually square shaped and have some physical extend. As such, the value of a pixel depends on the energy that arrives on the cell at a given time. Thus, the output of such a cell is some average value over the 3D region corresponding to the pixel: from this point of view the cell behaves like an integrator (in space and time).
- Second, the grey level values have no intrinsic meaning in general. The response of the CCD cell is usually not linear. Ideally, some calibration procedure should also be done for the grey level values, but since most cameras perform some automatic gain control, such a process is almost impossible to achieve if this gain control cannot be switched off in some way. The problem is even

worse with sequences of images since the gain may change from one image to the next, which has a consequence on the intensity derivative with respect to time!

- Finally, the grey level associated to a 3D region might change with the illumination and unless this effect is taken into account (or not considered by hypothesis for some reason), grey level values are not reliable data to work with.

Thus, some image formation characteristics might have important effects on the algorithms working directly on the grey level values. And the situation is actually even worse than that depicted since due to some technical problems, the camera signal is transformed many times before it is accessible from the computer. Fortunately, an important part of computer vision work is more based on some patterns or discontinuities of the grey levels than on their actual values, and is thus not directly affected by these problems. This, however, might not be the case for some shape from shading, some correlation or some optical flow techniques. For all these reasons, we have prefered to work with edges as primitives rather than with grey level values: this obviously has some limitations since only a small portion of the images can be well described by edge contours. However, it grounds our work on a firm basis from which we can (more or less) safely proceed.

3.4 Edges

3.4.1 What are Edges?

Edges are one of the most reliable information in images. They correspond to discontinuities in the grey level image $I(u, v)$. Such discontinuities are usually preserved under all the transformations that affect the grey level values as well as under the affine transformation due to the camera coordinate system.

However, such discontinuities may arise from many different situations. Basically, edges can be classified in two main categories:

Real edges arise from 3D physical discontinuities or singularities of the surface of the viewed object. What is important with this kind of edges is that they are

somewhat independent of the viewing geometry: as long as such an edge is seen in an image, it always refers to the same set of 3D points independently of the camera or of the light sources positions. This category of edge can be further refined in:

- Discontinuities of the normal to the object surface.
- Discontinuous change of the reflectance of the object.

Virtual edges are consequences more of the viewing geometry than of any real 3D discontinuities. The 3D points associated to such edges depend on the camera and light sources positions. Thus, without further assumptions, the structure can only, in theory, be recovered by differential techniques or by taking into account the reflectance properties of the 3D world. This leads to difficulties with stereo or structure from discrete motion. This type of edges includes:

- Discontinuities in the depth function $Z(u,v)$ that associates to each pixel the depth of the 3D point that is associated to it. Even in the absence of any other kind of discontinuities — which is generally not the case since —, this depth discontinuities give rise to edges due to their relative position to the light sources. One important instance of this kind of edges is occluding edges which are edges generated by the 3D surface points at which the optic ray is tangent to the surface. This case is quite important for close views of smooth objects and has been studied in [Vai90b, Vai90a, BC90].
- Shadow lines induced by obstacles between light sources and the viewed surface. The edge points depend on the position of the light sources with respect to the observed surface. Thus, as long as the cameras do not disturb the measurements in a stereo setup (i.e. each camera is not between the light source and the surface) or if light sources do not vary with time in a motion setup, these edges behave like real edges. This is not, however, the general case.

Of course, in general, an edge is due to a combination of these different causes. It is important, however, in some given environment, to be able to distinguish between real edges and virtual ones. We assume, hereafter, if not stated otherwise, that such a discrimination has been done so that only the real edges are considered in the image.

3.4.2 Edge Detection

Standard edge detection techniques are described in [Hil83a, Can86, Der87] but many more can be found in the literature. Usually, the basic idea is to detect the image discontinuities

- Either by looking at the maxima of the norm of the gradient of the image in the direction of the gradient.
- Or by searching for the zero crossings of the Laplacian of the image as presented in [Hil83a].

Of course, one of the main problems when extracting edges is the noise. In order to obtain reliable derivatives, a smoothing operation must be combined within the derivation process. Many different filters have been designed to achieve this goal each of which optimizes some aspect of edge detection. Among them, two techniques are widely used:

- [Can86] proposed a filter of finite extent designed to optimize a quality criterion that takes into account detection, localization of the edge and uniqueness of response. [Der87] has done the same work for a recursive filter. However, these criteria were derived only for 1D edges and then extended on 2D images. Thus, usually these filters have an anisotropic behavior when applied to 2D images (isotropic versions of these filters are neither separable nor recursive and are thus very expensive to compute).
- Gaussian filters have the main advantage of being separable and still isotropic with 2D images. It has moreover some interesting properties when dealing with scale space [AGLM92, KKvD93]. Implementing such a filter is however not an easy task since its support is infinite. Classically, two methods are used. Either the convolution is made in image space using a finite kernel or in the Fourier domain. Both methods have advantages and drawbacks. However Deriche have proposed recently a method that combines the nice properties of the Gaussian with the efficiency of recursive filters: the basic idea is to approximate the Gaussian or its derivatives by a recursive filter of the appropriate form [Der92, Der93].

These edge extraction operations are then completed with a postprocessing that eliminates the smallest edges and links edge pixels into *chains*.

In the work presented here, we have used two kinds of edge detectors. The first is a Canny-Deriche that gives integer coordinates for the edge pixels. The second one uses the recursive implementation of the first order derivatives of the Gaussian for the filtering step. Then, non maxima suppression, hysteresis thresholding and linking are performed just like with the Canny-Deriche but the edge extraction is done with subpixelic accuracy. The rationale behind subpixel edge detection is that pixels have some physical extent. Thus, at an edge the grey level value is intermediate between the grey values on both sides of the edge. Obviously the problems that were mentioned in the previous sections must appear here, but using the method described in section 5.4, it has been proved that the method we use gives edge points that are accurate to half or a quarter of a pixel with real images. Thus since accuracy appeared to be crucial for our purpose, this last method was usually prefered.

Remark 3.6 *Notice also that in all the methods we have used for detecting edges, the extraction was done using image coordinates. Thus, when extracting the edges along the gradient direction, we encounter the problem mentioned at section 3.1.2. However, although ideally one have to compensate for it, this effect might not be as important for edge detection as for many other problems. This is because maxima are a topological information of the intensity surface so that, even a quite important error in the orientation of the direction along which the maxima are extracted, does not affect much the actual position of the resulting edge.*

Chapter 4

Optical Flow and Motion Field: the Curve Case

This chapter introduces the main interest of this work which is the problem of motion and structure. Normally, the right place for this chapter would have been as an introduction to part III. However, it is placed here since it introduces all the quantities that we will need and since the computation of those is the topic of the next part. Its main achievement is the full study of the motion field of the image curve generated by a moving non-elastic 3D curve. It defines two different motion fields and show clearly what components of these fields can be recovered from an image sequence. As such, the work described here provides the ground on which all the subsequent study of the motion of curves is based. We also show in this chapter some image sequences designed to try to understand better how the biological vision systems solve the problems that are raised by the results shown in this chapter.

4.1 Optic flow and Motion Field

The concept of optical flow was introduced by Gibson [Gib50] and is based upon the idea that there is a relationship between the temporal variations of the image

intensity at one point of the retinal plane and the motion of the camera, the motions and the shapes of objects present in the scene. As we noticed it already some photometric model of the scene would, in theory, be necessary to achieve such a task.

4.1.1 The Optical Flow

Let us consider the image intensity $I(\mathbf{m},\tau)$ at pixel $\mathbf{m}$ in the retinal plane at time τ. $\mathbf{m}$ is the image of a 3D point $\mathbf{M}$ moving in the scene with a velocity $\mathbf{V_M}$. The velocity of $\mathbf{m}$ is $\mathbf{v_m}$ (see figure 4.1). If we take the total time derivative $\dot{I}$ of I with respect to time, we obtain:

$$\dot{I} = \nabla I \cdot \mathbf{v_m} + \frac{\partial I}{\partial \tau} \ .$$

This formula involves no approximations but involves one quantity, $\dot{I}$, which cannot be computed simply from the sequence of images. Actually, in order to compute it, we need to introduce models of the scene reflectance. However, the standard optical flow constraint is defined by assuming that $\dot{I} = 0$:

$$\nabla I \cdot \mathbf{v_m} + \frac{\partial I}{\partial \tau} = 0 \ , \tag{4.1}$$

which has been presented by many authors as a constraint on the velocity $\mathbf{v_m}$. This constraint is the so-called motion constraint equation. It is easy to see, however, that the assumption that $\dot{I} = 0$ does not hold even with very simple photometric models (see [FP93] for a simple example, or [VP87] for a more complex examples). Consequently, equation (4.1) should *not* be considered as a constraint on the image velocity field $\mathbf{v_m}$, but as the *definition* of a new image vector field $\mathbf{v^o_m}$, the optical flow, parallel to ∇I

$$\mathbf{v^o_m} = -\frac{\frac{\partial I}{\partial \tau}}{\|\nabla I\|} \frac{\nabla I}{\|\nabla I\|} \ . \tag{4.2}$$

The qualitative properties of this vector field are similar to those of the motion field $\mathbf{v_m}$ so that, tasks like segmentation from motion or like giving qualitative properties of the motions can still be performed [VGT89, BF93]. It is difficult, however, to use this information in a quantitative manner as for example for the problem of structure from motion.

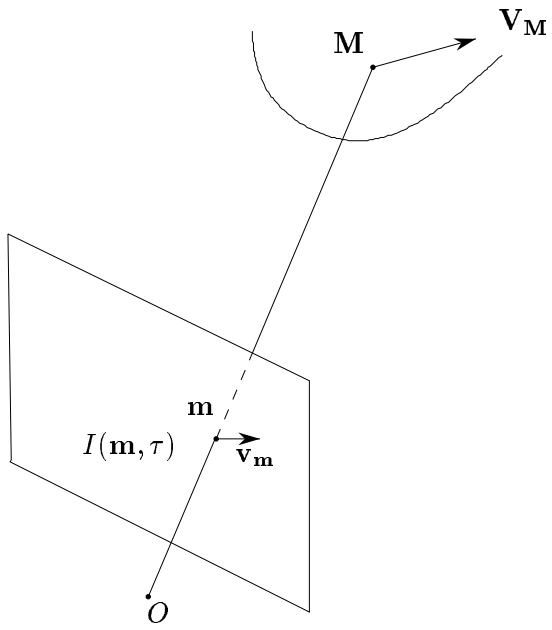

Figure 4.1: Optical flow.

Remark 4.1

- *Taking the total time derivative of $I(\mathbf{m},\tau)$ supposes that I is a differentiable function with respect to time which is not the case at the edges since those correspond to discontinuities of $I(\mathbf{m},\tau)$.*
- *Equation (4.1) is in theory valid on surfaces defined by $I(\mathbf{m},\tau) = constant$. However, it is known that edges are isointensity curves only in a first approximation and considering the spatio-temporal situation worsens the quality of this approximation. It is thus dangerous to give any spatio-temporal meaning to isointensity surfaces.*

Following Horn [Hor86], we draw a sharp distinction between the motion field $\mathbf{v_m}$ and the optical flow field $\mathbf{v}^o_\mathbf{m}$. Note that $\mathbf{v}^o_\mathbf{m}$ can be computed from a sequence of images.

4.1.2 The Motion Field

According to the previous discussion, the motion field $\mathbf{v_m}$ is the time derivative $\dot{\mathbf{m}}$ of the representation $\mathbf{m}$ of pixel m. This quantity can be derived from the perspective equation (3.1) in the case of a rigid 3D motion. Taking the total time derivative of this equation leads to

$$\mathbf{V}_{\mathbf{M}_Z}\mathbf{m} + Z\mathbf{v_m} = \mathbf{V_M} \ ,$$

where $\mathbf{V_M}$ is the 3D velocity of point $\mathbf{M}$. From section 3.2.2, we know that

$$\mathbf{V_M} = \mathbf{V} + \mathbf{\Omega} \wedge \mathbf{M} \ .$$

Thus, we obtain:

$$Z(\mathbf{v_m} - \mathbf{\Omega} \wedge \mathbf{m} + (\mathbf{\Omega}, \mathbf{m}, \mathbf{k})\mathbf{m}) = \mathbf{V} - \mathbf{V}\cdot\mathbf{k}\,\mathbf{m} \ . \tag{4.3}$$

If we have several n pixels $\mathbf{m}_1, \cdots, \mathbf{m}_n$ at which we know the motion fields $\mathbf{v}_{\mathbf{m}_1}, \cdots, \mathbf{v}_{\mathbf{m}_n}$ and if we assume that the n corresponding 3D points $\mathbf{M}_1, \cdots, \mathbf{M}_n$ belong to the same object moving rigidly with the kinematic screw $(\mathbf{\Omega}, \mathbf{V})$, we have

$2n$ equations in the unknowns $(\mathbf{\Omega}, \mathbf{V}, Z_1, \cdots, Z_n)$. To solve for those unknowns is to solve the so-called *motion and structure* problem. A closer look at equation (4.3) shows that the problem is homogeneous in $(\mathbf{V}, Z_1, \cdots, Z_n)$, therefore the total number of unknowns is $n + 5$.

A naive counting of the equations tells us that if n is larger than or equal to 5, we may be able to solve the problem[1]. Note that the equations are linear in $\mathbf{\Omega}$ but nonlinear in $(\mathbf{V}, Z_1, \cdots, Z_n)$.

But as it was discussed previously, it turns out that the most difficult task is *not* to solve this system of nonlinear equations but to actually compute the motion field $\mathbf{v_m}$ with enough accuracy at a sufficiently large number of points. At edges, the well known aperture problem states that only the normal component of the velocity field can be recovered. So full velocities can be recovered only at special points that can be tracked (such as inflexion or bitangent points or special grey level patterns that are known to be preserved in some way under perspective projection).

The standard approach to this problem in computer vision has been so far to identify $\mathbf{v}^o_\mathbf{m}$, the optical flow field defined by equation (4.2) with the projection of the motion field $\mathbf{v_m}$ along the direction of the gradient

$$\mathbf{v}^o_\mathbf{m} = \left(\mathbf{v_m} \cdot \frac{\nabla I}{\|\nabla I\|}\right) \frac{\nabla I}{\|\nabla I\|} \,. \tag{4.4}$$

As we have already seen it, this equation results from assumptions that are not generally true.

Assuming this equation, this yields a linear constraint on the two coordinates of $\mathbf{v_m}$ and therefore only one equation at every pixel, obtained by projecting the two sides of (4.4) along the direction of the image gradient. The motion and structure problem is thus impossible to solve since n points yield n equations in $n+5$ unknowns. In order to overcome this new problem, researchers have tried to "invent" a motion field by imposing a smoothness constraint [Hor86, Hil83b, WW88, Nag83].

The first kind of idea is to find a smooth field $\hat{\mathbf{v}}_\mathbf{m}$ whose component along the direction of the image gradient is equal to the measured optical flow $\mathbf{v}^o_\mathbf{m}$. Practically,

[1] That this is indeed the case and that the number of solutions is at most ten has been proved by Maybank [May90b]

however, this cannot be done exactly and an optimization problem that minimize the difference between the projection of the field $\hat{\mathbf{v}}_\mathbf{m}$ onto the gradient direction and the measured optical flow $\mathbf{v}^o_\mathbf{m}$. This can be expressed mathematically as the following minimization problem:

$$\min_{\mathbf{v_m}} \int\int ((\mathbf{v_m} \cdot \frac{\nabla I}{\|\nabla I\|} - \mathbf{v}^o_\mathbf{m})^2 + \lambda Tr(D\mathbf{v_m}(D\mathbf{v_m})^T))dxdy \ , \tag{4.5}$$

where $D\mathbf{v_m}$ is the Jacobian of the function $\mathbf{v_m}(\mathbf{m})$ with respect to the (spatial) coordinates of $\mathbf{m}$ at the pixel $\mathbf{m}$. Denoting these coordinates as $\mathbf{v_m} = [v_x, v_y]$, we have:

$$D\mathbf{v_m} = \begin{bmatrix} \frac{\partial v_x}{\partial x} & \frac{\partial v_x}{\partial y} \\ \frac{\partial v_y}{\partial x} & \frac{\partial v_y}{\partial y} \end{bmatrix} .$$

The criterion (4.5) is the sum of two terms: the first term imposes that the component of the "invented" field $\hat{\mathbf{v}}_\mathbf{m}$ along the gradient direction is as close as possible to the measurements $\mathbf{v}^o_\mathbf{m}$ and the second term controls, through the parameter λ, its smoothness.

A detailed analysis of the possible solutions to this minimization problem can be found in [Hor86]. For a related approach, see [Hil83b] or [Nag83]. Of course there is no guarantee that the "invented" field is close to the motion field $\mathbf{v_m}$ and in fact it is, in general, different. We show this in Appendix C for a method related to the one described in [Hil83b].

Another kind of approaches include those of Lucas and Kanade [LK81], Waxman and Wohn [WW88], Fleet and Jepson [FJ90] and Singh [Sin90] that fit the measured normal flows to a local model of the 2D velocity field (e.g. a low order polynomial model). Finally, some methods [Nag83, UGVT88] use the first order derivatives (spatial and/or temporal) of equation (4.1) to get more constraints on the optical flow. Notice that some of these techniques are difficult to use along edges since at such points the available information is essentially one dimensional.

Before continuing, let us derive a useful formula that will be used in the remaining of this text. This formula relates the component of the velocity field in one specific

direction $\mathbf{l}$ (at this point $\mathbf{l}$ can be any vector). So let us project equation (4.3) onto vector $\mathbf{l}$

$$Z(\mathbf{v_m}\cdot\mathbf{l} - (\mathbf{\Omega}, \mathbf{m}, \mathbf{l} - (\mathbf{m}\cdot\mathbf{l})\mathbf{k})) = \mathbf{V}\cdot(\mathbf{l} - (\mathbf{m}\cdot\mathbf{l})\mathbf{k}) \; .$$

Since $\mathbf{m}\cdot\mathbf{k} = 1$, we recognize that $\mathbf{l} - (\mathbf{m}\cdot\mathbf{l})\mathbf{k} = (\mathbf{m}\cdot\mathbf{k})\mathbf{l} - (\mathbf{m}\cdot\mathbf{l})\mathbf{k} = \mathbf{m} \wedge (\mathbf{l} \wedge \mathbf{k})$ and thus obtain:

$$Z(\mathbf{v}_r\cdot\mathbf{l} - \mathbf{\Omega}\cdot(\mathbf{m} \wedge (\mathbf{m} \wedge (\mathbf{l} \wedge \mathbf{k})))) = \mathbf{V}\cdot(\mathbf{m} \wedge (\mathbf{l} \wedge \mathbf{k})) \; . \tag{4.6}$$

Notice that equation (4.6) is true for every point $\mathbf{m}$ of the retina and for every vector $\mathbf{l}$. Moreover, the third component of equation (4.3) is trivially true (do not forget that $\mathbf{m}\cdot\mathbf{k} = 1$), which means that equation (4.6) is totally independent of the third coordinate of $\mathbf{l}$. As a consequence, we can restrict $\mathbf{l}$ to be parallel to the retina $\mathcal{R}$ without loss of generality.

In the remaining of this chapter, we are going to investigate in great depth the relationship between $\mathbf{v_m}$ and $\mathbf{V_M}$ along edges. Then in part III, we will make use of the kinematic equations introduced in this section so as to derive a new formulation for the motion and structure problem for rigid curves.

4.2 Spatio-temporal Surfaces

We now assume that we observe in a sequence of images a family (c_τ) of curves where τ denotes the time, which we assume to be the perspective projection in the retina of a 3D curve (C) that moves in space. If we consider the three-dimensional space (x, y, τ), this family of curves sweeps in that space a surface (Σ) defined as the set of points $((c_\tau), \tau)$. As an example, figure 4.2 shows the spatio-temporal surface generated by a circle rotating around one of its diameters in front of the camera.

This surface encodes all the information available from the image sequence about the 3D curve. Consequently, the study of this surface is of great importance for the understanding of the motion-and-structure problem for curves. The next two sections are dedicated to this task.

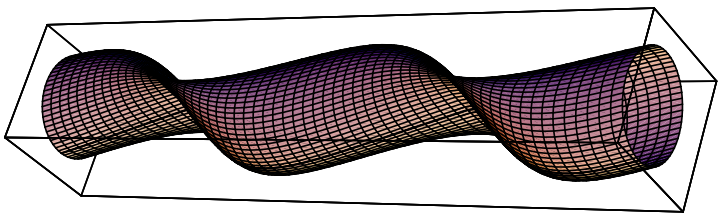

Figure 4.2: The spatio-temporal surface generated by a circle rotating in front of the camera.

4.3 Real versus Apparent Motion Field

At a given time instant τ, let us consider the observed curve (c_τ). Its arclength s can be computed and (c_τ) can be parameterized by s and τ: it is the set of points $\mathbf{m}(s,\tau)$ in the retinal plane. The corresponding points $\mathbf{P}$ on (Σ) are represented by the vector $\mathbf{P} = (\mathbf{m}^T(s,\tau),\tau)^T$. Now, let S be an arclength defined along the 3D curve (C). We assume that the motion of (C) preserves the arclength. This rules out elastic motions but allows rope-like and rigid motions. Such motions are called *isometric* motions. Then, notice that the arclength s of (c_τ) is a function $s(S,\tau)$ of the arclength S of the 3D curve (C) and of the time τ, and that the two parameters (S,τ) can also be used to parameterize (Σ) in a neighborhood of $\mathbf{P}$. Of course, the function $s(S,\tau)$ is unknown.

As shown in figure 4.3, we can consider on (Σ) the curves defined by $s = constant$ or $S = constant$. These curves are in general different, and their projections, parallel to the τ-axis, in the (x,y)-plane have an important physical interpretation, related to our upcoming definition of the motion fields.

Indeed, as shown in figure 4.4, suppose we choose a point $\mathbf{M}_0$ on (C) and fix its arclength S_0 at time τ. When (C) moves, this point follows a trajectory $(C_{\mathbf{M}_0})$ in 3D-space and its image $\mathbf{m}_0$ follows a trajectory $(c^r_{\mathbf{m}_0})$ in the retinal plane. This last curve is the projection in the retinal plane, parallel to the τ-axis, of the curve defined by $S = S_0$ on the surface (Σ). We call it the "real" trajectory of m_0 because it is the retinal image of $(C_{\mathbf{M}_0})$ which is the trajectory of a physical point attached to (C).

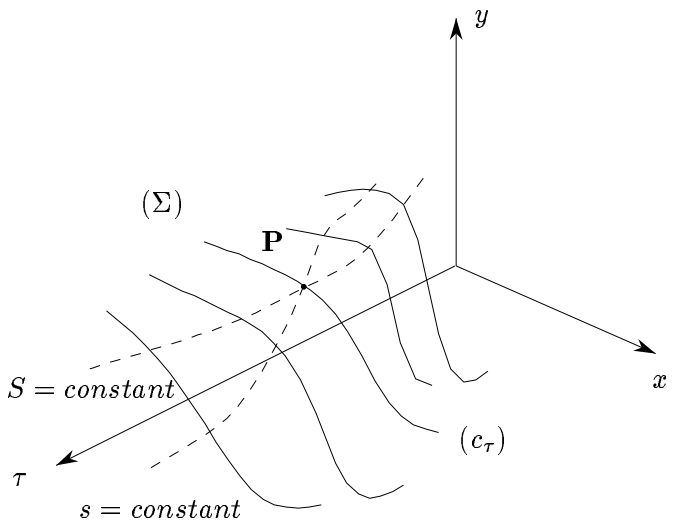

Figure 4.3: Definition of the spatio-temporal surface (Σ).

We can also consider the same projection of another curve defined on (Σ) by $s = s_0$. The corresponding curve $(c^a_{\mathbf{m}_0})$ in the retinal plane is the trajectory of the image point $\mathbf{m}_0$ of arclength s_0 on (c_τ). We call this curve the "apparent" trajectory of $\mathbf{m}_0$ (see figure 4.5) because it is not, in general, the retinal image of a physical point attached to (C).

The mathematical reason why those two curves are different is that the first one is defined by $S = S_0$ while the second is defined by $s(S, \tau) = s_0$.

Let us now define precisely what we mean by motion fields. If we consider figure 4.6, point $\mathbf{m}$ on (c_τ) is the image of point $\mathbf{M}$ on (C). This point has a $3D$ velocity $\mathbf{V_M}$ whose projection in the retina is the *real motion field* $\mathbf{v}^r_\mathbf{m}$ (r for *real*); mathematically speaking:

- $\mathbf{v}^r_\mathbf{m}$ is the partial derivative of $\mathbf{m}(s, \tau)$ with respect to time when S is kept constant, or its total time derivative $\dot{\mathbf{m}}$ (it is the vector $\mathbf{v_m}$ of the previous section).
- The *apparent motion field* $\mathbf{v}^a_\mathbf{m}$ (a for *apparent*) of $\mathbf{m}(s, \tau)$ is the partial derivative with respect to time when s is kept constant, $\frac{\partial \mathbf{m}}{\partial \tau} = \mathbf{m}_\tau$.

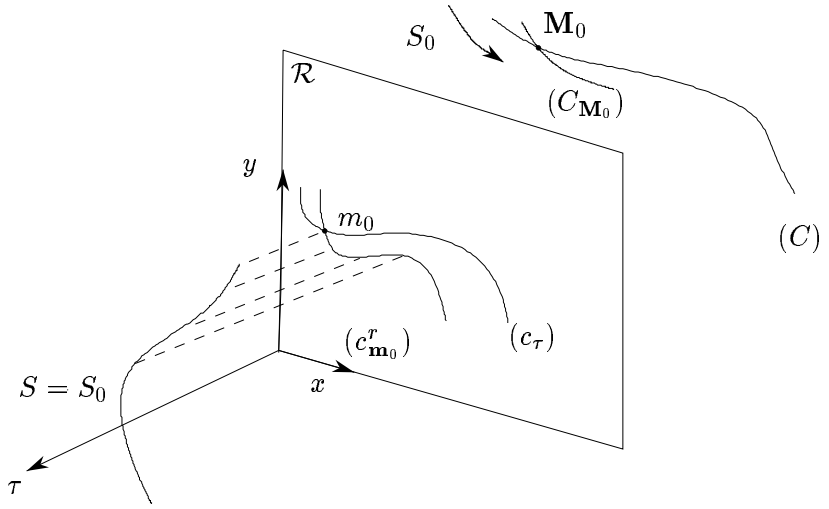

Figure 4.4: Projection in the image plane, parallel to the τ-axis, of the curve $S = S_0$ of the surface (Σ): $(c^r_{\mathbf{m}_0})$ is the "real" trajectory of $\mathbf{m}_0$.

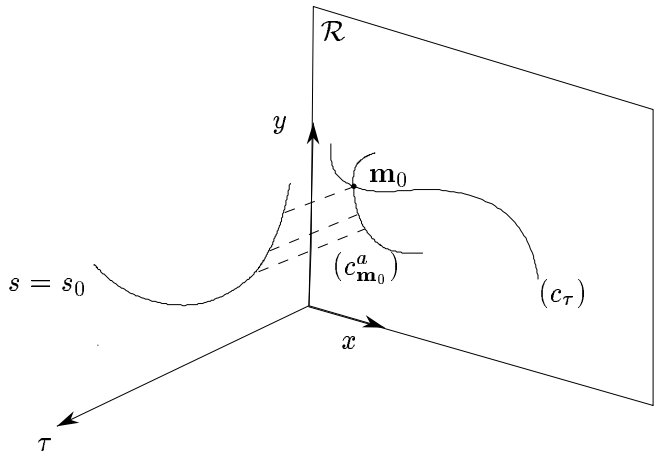

Figure 4.5: Projection in the image plane, parallel to the τ-axis, of the curve $s = s_0$ of the surface (Σ): $(c^a_{\mathbf{m}_0})$ is the "apparent" trajectory of $\mathbf{m}_0$.

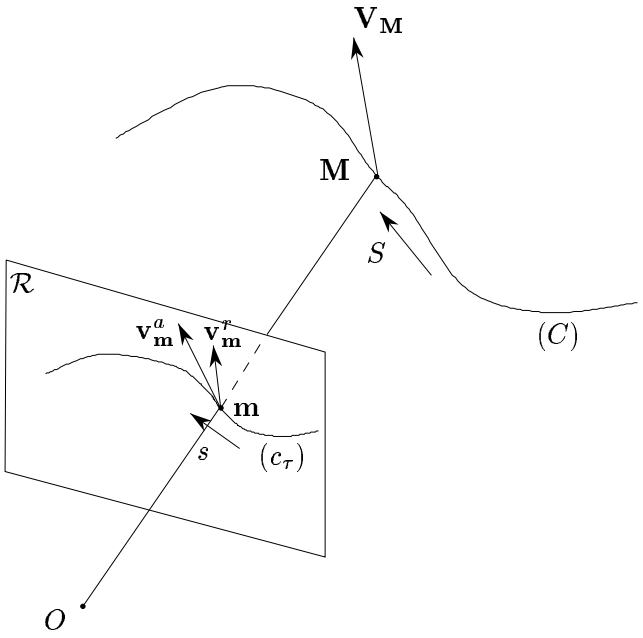

Figure 4.6: Definition of the two motion fields: the real and the apparent.

Those two quantities are in general distinct. To relate this to the previous discussion about the curves $S = S_0$ and $s = s_0$ of (Σ), the vector $\mathbf{v}^a_\mathbf{m}$ is tangent to the "apparent" trajectory of $\mathbf{m}$, while $\mathbf{v}^r_\mathbf{m}$ is tangent to the "real" one. This is summarized in figure 4.7. Figure 4.8 shows how much the two fields $\mathbf{v}^r_\mathbf{m}$ and $\mathbf{v}^a_\mathbf{m}$ are different.

We now make the following important remark. All the information about the motion of points of (c_τ) (and of the 3D points of (C) which project onto them) is entirely contained in the surface (Σ). Since (Σ) is intrinsically characterized, up to a rigid motion, by its first and second fundamental forms and the Gauss and Mainardi-Codazzi equations (see section 2.4), they are all we need to characterize the motion fields of (c_τ) and the motion of (C). Our main conclusion will be that only the apparent motion field can be recovered from (Σ).

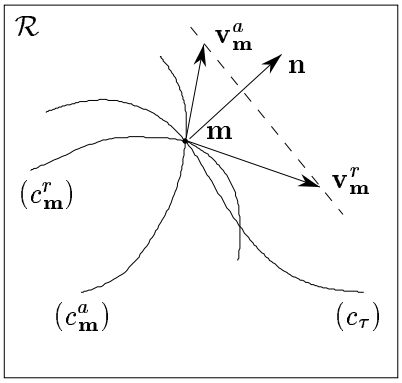

Figure 4.7: Comparison of the two motion fields and the real and apparent trajectories: $\mathbf{n}$ is the normal to (c_τ).

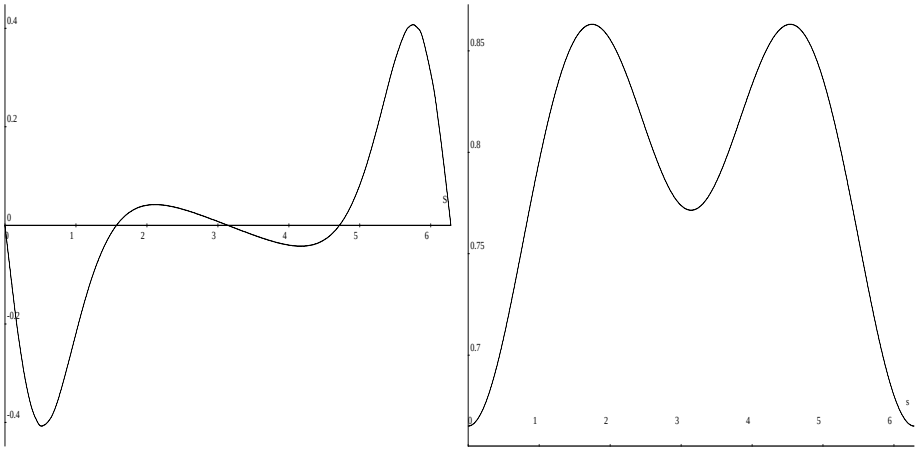

Figure 4.8: A plot of the real (left) and apparent (right) motion fields along an ellipse. The horizontal axis is the arclength, the vertical one is the value of the tangential field.

4.4 Characterizing the Spatio-Temporal Surface

In this section, we compute the first and second fundamental forms and the Mainardi-Codazzi equations of the spatio-temporal surface (Σ). On the way, we shall be gleaning a number of interesting facts relative to the motion and deformation of the curve (c_τ). The following result about functions defined on (c_τ) and thus on (Σ) will often be used hereafter.

Given a function f of the variables s and τ, it is a function on (c_τ). It is also a function f' of S and τ, and therefore it also defines a function on (Σ). We will have to compute $\frac{\partial f'}{\partial S}$ and $\frac{\partial f'}{\partial \tau}$. The second derivative is also called the total time derivative of f with respect to time, $\dot{f}$; introducing $u = \frac{\partial s}{\partial S}$ and $v = \frac{\partial s}{\partial \tau}$, we have the following equations:

$$\begin{array}{rcccccl} f'_S & = & \frac{\partial f'}{\partial S} & = & u\frac{\partial f}{\partial s} & = & uf_s\ , \\ \dot{f} & = & \frac{\partial f'}{\partial \tau} = f'_\tau & = & v\frac{\partial f}{\partial s} + \frac{\partial f}{\partial \tau} & = & vf_s + f_\tau\ . \end{array} \tag{4.7}$$

Note that when we write $\frac{\partial f'}{\partial \tau}$ in the second set of equations (4.7), it is a partial derivative at $S = constant$ whereas when we write $\frac{\partial f}{\partial \tau}$, it is a derivative at $s = constant$.

Following these notations, we denote by $\mathbf{P}(s,\tau) = [\mathbf{m}^T(s,\tau),\tau]^T$ the generic point of (Σ) and by $\mathbf{P}'(S,\tau) = [\mathbf{m}'^T(S,\tau),\tau]^T$ the same point considered as a function of S and τ.

4.4.1 The First Fundamental Form of the Spatio-Temporal Surface (Σ)

Using equations (4.7), and the first two-dimensional Frenet formula, we write

$$\mathbf{P}'_S = u\mathbf{P}_s = u\begin{bmatrix} \mathbf{m}_s \\ 0 \end{bmatrix} = \begin{bmatrix} u\mathbf{t} \\ 0 \end{bmatrix}\ , \tag{4.8}$$

$$\mathbf{P}'_\tau = v\mathbf{P}_s + \mathbf{P}_\tau = v\begin{bmatrix} \mathbf{m}_s \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{m}_\tau \\ 1 \end{bmatrix} = \begin{bmatrix} v\mathbf{t} + \mathbf{v}^a_{\mathbf{m}} \\ 1 \end{bmatrix}\ , \tag{4.9}$$

in which $\mathbf{t}$ is the unit tangent vector to (c_τ) at $\mathbf{m}$.

We now write the apparent motion field $\mathbf{v}^a_\mathbf{m}$ in the Frenet frame $\mathbf{t}, \mathbf{n}$, where $\mathbf{n}$ is the unit normal vector to (c_τ) at $\mathbf{m}$:

$$\mathbf{v}^a_\mathbf{m} = \alpha\mathbf{t} + \beta\mathbf{n} \ . \tag{4.10}$$

We call α and β the tangential and normal apparent motion fields, respectively. We see from equation (4.9) that $\mathbf{P}'_\tau = [(v+\alpha)\mathbf{t}^T + \beta\mathbf{n}^T, 1]^T$; but by definition, $\mathbf{P}'_\tau = [\mathbf{m}'^T_\tau, 1]^T = [\dot{\mathbf{m}}^T, 1]^T = [\mathbf{v}^{rT}_\mathbf{m}, 1]^T$. Therefore the quantity $v+\alpha$ is the tangential real motion field which we denote by w and β is the normal real motion field. The real and apparent motion fields have the same component along $\mathbf{n}$, we call it the normal motion field (see figure 4.7). According to all this, the real motion field is given by:

$$\mathbf{v}^r_\mathbf{m} = w\mathbf{t} + \beta\mathbf{n} \ . \tag{4.11}$$

Let us define $\mathbf{V}^a_\mathbf{m}$ as $[\mathbf{v}^{aT}_\mathbf{m}, 1]^T$ and $\mathbf{V}^r_\mathbf{m}$ as $[\mathbf{v}^{rT}_\mathbf{m}, 1]^T = \mathbf{P}'_\tau$. $\mathbf{V}^a_\mathbf{m}$ and $\mathbf{V}^r_\mathbf{m}$ are three-dimensional vectors. Consider the tangent plane $\mathbf{T}_P$ to the spatio-temporal surface (Σ) at $\mathbf{P}$. By definition, it is spanned by the two vectors $\mathbf{P}'_S$ and $\mathbf{P}'_\tau$ (see section 2.4). Examining those two vectors, we see that the vectors $\mathbf{t}_0 = [\mathbf{t}^T, 0]^T$ and $\mathbf{n}_\beta = [\beta\mathbf{n}^T, 1]^T$ which are orthogonal, also span $\mathbf{T}_P$ since $\mathbf{P}'_S = u\mathbf{t}_0$ and $\mathbf{P}'_\tau = \mathbf{V}^r_\mathbf{m} = w\mathbf{t}_0 + \mathbf{n}_\beta$ (see figure 4.9). From this follows that the two vectors $\mathbf{V}^r_\mathbf{m} = w\mathbf{t}_0 + \mathbf{n}_\beta$ and $\mathbf{V}^a_\mathbf{m} = \alpha\mathbf{t}_0 + \mathbf{n}_\beta$ belong to $\mathbf{T}_P$ and define on (Σ) two tangent vector fields. The relationship between those vectors of $\mathbf{T}_P$ is shown in figure 4.10.

Expanding on this idea, we can give a geometric interpretation of the operation of partial derivative $\frac{\partial}{\partial\tau}$ when the image arclength s is kept constant and of the total time derivative. Given a function f of s and τ into R, it induces a function F from (Σ) by $f(s,\tau) = F(\mathbf{P}(s,\tau))$. $\frac{\partial f}{\partial\tau}$ is the directional derivative of F in $\mathbf{T}_P$ along $\mathbf{V}^a_\mathbf{m}$ which we denote by $L_{\mathbf{V}^a_\mathbf{m}}F$. This is called the Lie derivative of the function F with respect to the tangent field $\mathbf{V}^a_\mathbf{m}$ (see section 2.4). It satisfies the following linear property:

$$L_{\mathbf{V}^a_\mathbf{m}} = L_{\alpha\mathbf{t}_0+\mathbf{n}_\beta} = \alpha L_{\mathbf{t}_0} + L_{\mathbf{n}_\beta} \ .$$

Thus

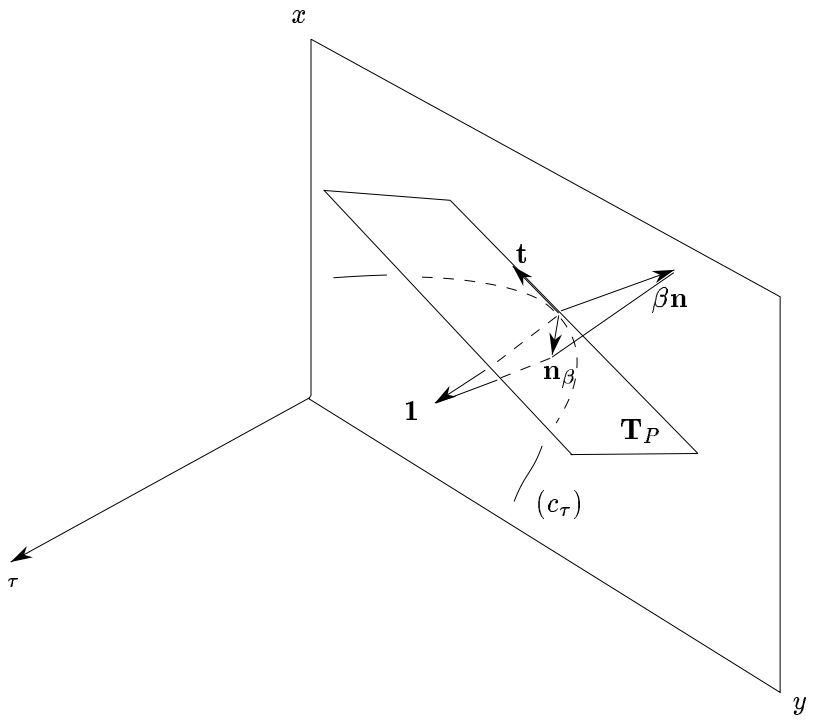

Figure 4.9: The vectors $\mathbf{t}_0$ and $\mathbf{n}_\beta$ span the tangent plane $\mathbf{T}_P$ to Σ.

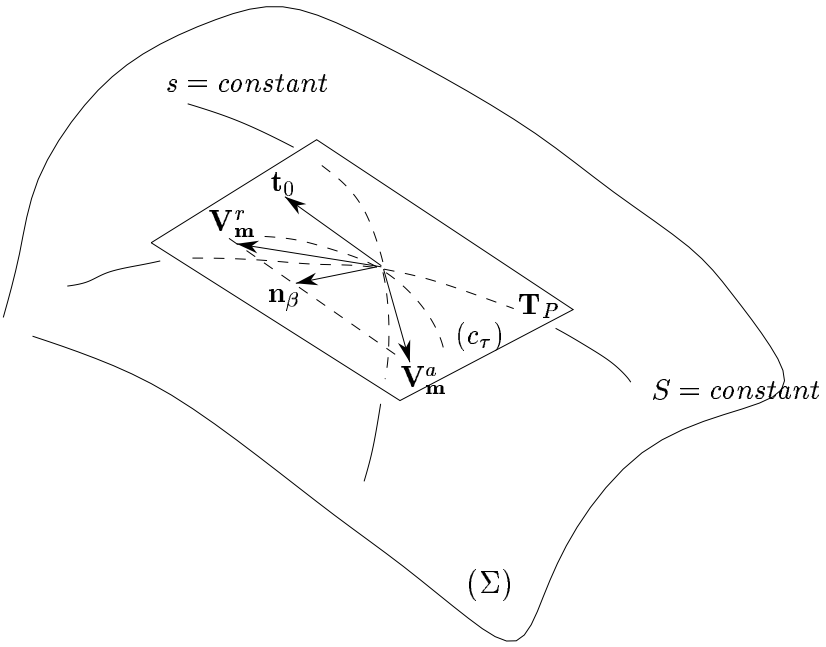

Figure 4.10: Various vectors of the tangent plane $\mathbf{T}_P$ to (Σ).

$$\frac{\partial f}{\partial \tau} = \alpha L_{\mathbf{t}_0} F + L_{\mathbf{n}_\beta} F \ .$$

$L_{\mathbf{t}_0} F$ is simply $\frac{\partial f}{\partial s}$ and we note $L_{\mathbf{n}_\beta} F = \partial_{\mathbf{n}_\beta} f$. The meaning of this quantity is shown in figure 4.11. We consider the normal $\mathbf{n}$ at the point $\mathbf{m}$ of the curve (c_τ). At time $\tau + d\tau$, the curve $(c_{\tau+d\tau})$ is intersected by the line defined by $\mathbf{m}$ and $\mathbf{n}$ at a point represented by $\mathbf{m} + \beta \mathbf{n} d\tau$ and we have

$$\partial_{\mathbf{n}_\beta} f = \lim_{d\tau \to 0} \frac{f(\mathbf{m} + \beta \mathbf{n} d\tau) - f(\mathbf{m})}{d\tau} \ .$$

Thus we write:

$$\frac{\partial f}{\partial \tau} = \alpha \frac{\partial f}{\partial s} + \partial_{\mathbf{n}_\beta} f \ . \tag{4.12}$$

Similarly, the total time derivative $\dot{f}$ of f, is the directional derivative of F in $\mathbf{T}_P$ along $\mathbf{V}^r_{\mathbf{m}}$:

$$\dot{f} = L_{\mathbf{V}^r_{\mathbf{m}}} F = w L_{\mathbf{t}_0} F + L_{\mathbf{n}_\beta} F = w \frac{\partial f}{\partial s} + \partial_{\mathbf{n}_\beta} f \ . \tag{4.13}$$

Equations (4.12) and (4.13) have the advantage of expressing $\frac{\partial f}{\partial \tau}$ and $\dot{f}$ as functions of $\frac{\partial f}{\partial s}$ and $\partial_{\mathbf{n}_\beta} f$ which can be computed from the sequence of curves (for $\partial_{\mathbf{n}_\beta} f$ we need to know β but we will show in a moment that it can be estimated from (Σ)), and of the two unknown tangential motion fields, the apparent one α, and the real one w.

Equations (4.12) and (4.13) also hold for functions f into R^p. We will be using heavily the cases $p = 2, 3$ in what follows.

From equations (4.8), (4.9), and equations (2.5), we can compute the coefficients of the first fundamental form (see section 2.4):

Proposition 4.1 *The coefficients of the first fundamental form in the basis* $(\mathbf{P}'_S, \mathbf{P}'_\tau)$ *of* $\mathbf{T}_P$ *are given by:*

$$E = u^2 \ , \quad F = uw \ , \quad G = 1 + w^2 + \beta^2 \ . \tag{4.14}$$

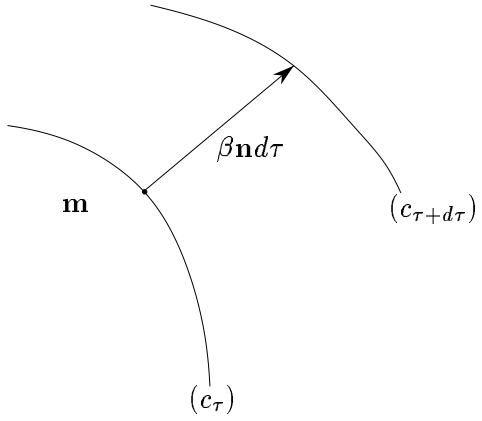

Figure 4.11: A geometric interpretation of $\partial_{\mathbf{n}_\beta}$.

We can also compute those coefficients in the basis $(\mathbf{t}_0, \mathbf{n}_\beta)$:

Proposition 4.2 *The coefficients of the first fundamental form in the basis* $(\mathbf{t}_0, \mathbf{n}_\beta)$ *of* $\mathbf{T}_P$ *are given by:*

$$E' = 1 \ , \quad F' = 0 \ , \quad G' = 1 + \beta^2 \ .$$

Proof: Let us denote φ the linear mapping $\mathbf{T}_P \rightarrow \mathbf{T}_P$ such that $\mathbf{\Phi}_1 \mathbf{x} = \varphi \mathbf{x} \cdot \mathbf{x}$ for all $\mathbf{x}$ of $\mathbf{T}_P$. Since we have:

$$\mathbf{t}_0 = \tfrac{1}{u} \mathbf{P}'_S \ , \quad \mathbf{n}_\beta = -\tfrac{w}{u} \mathbf{P}'_S + \mathbf{P}'_\tau \ , \tag{4.15}$$

we obtain immediately $E' = \mathbf{\Phi}_1 \mathbf{t}_0 = \frac{1}{u^2} \mathbf{\Phi}_1 \mathbf{P}'_S = \frac{E}{u^2} = 1$, $G' = \mathbf{\Phi}_1 \mathbf{n}_\beta = \frac{w^2}{u^2} E - 2 \frac{w}{u} F + G = 1 + \beta^2$, and $F' = \varphi \mathbf{t}_0 \cdot \mathbf{n}_\beta = -\frac{w}{u^2} E + \frac{1}{u} F = 0$.
□

A normal $\mathbf{N}_P$ to (Σ) that will be needed for the second fundamental form can also be computed:

$$\mathbf{N}_P = \mathbf{t}_0 \wedge \mathbf{n}_\beta = \begin{bmatrix} \mathbf{t} \\ 0 \end{bmatrix} \wedge \begin{bmatrix} \beta \mathbf{n} \\ 1 \end{bmatrix} \ .$$

Let $\mathbf{t} = [\mathbf{t}_x, \mathbf{t}_y]^T$, then $\mathbf{n} = \varepsilon[\mathbf{t}_y, -\mathbf{t}_x]^T$ where $\varepsilon = \pm 1$. The cross product appearing in the previous equation is therefore equal to

$$\begin{bmatrix} \mathbf{t}_y \\ -\mathbf{t}_x \\ -\varepsilon\beta \end{bmatrix} = \varepsilon \begin{bmatrix} \mathbf{n} \\ -\beta \end{bmatrix} .$$

Finally

$$\mathbf{N}_P = \varepsilon \begin{bmatrix} \mathbf{n} \\ -\beta \end{bmatrix} .$$

Given a normal $\mathbf{N}_P$ to the spatio-temporal surface (Σ) whose coordinates in the coordinate system $(\mathbf{t}, \mathbf{n}, \tau)$ (τ is the unit vector defining the τ-axis) are denoted by $N_\mathbf{t}, N_\mathbf{n}, N_\tau$, we have:

$$\beta = -\tfrac{N_\tau}{N_\mathbf{n}} , \quad N_\mathbf{t} = 0 .$$

We have thus proved the following proposition:

Proposition 4.3 *The normal to the spatio-temporal surface* (Σ) *yields an estimate of the* normal motion field β *as*

$$\beta = -\frac{N_\tau}{N_\mathbf{n}} . \tag{4.16}$$

In what follows, we take $\mathbf{N}_P = [\mathbf{n}, -\beta]^T$ since $\mathbf{N}_P$ is defined up to a scale factor.

4.4.2 The Second Fundamental Form of the Spatio-Temporal Surface (Σ)

We denote $\frac{\partial^2 \boldsymbol{s}}{\partial S^2}$ by u_S, $\frac{\partial^2 \boldsymbol{s}}{\partial \tau^2}$ by v_τ, $\frac{\partial^2 \boldsymbol{s}}{\partial \tau \partial S}$ by u_τ, and $\frac{\partial^2 \boldsymbol{s}}{\partial S \partial \tau}$ by v_S. We are going to compute $\frac{\partial^2 \mathbf{P}'}{\partial S^2}$, $\frac{\partial^2 \mathbf{P}'}{\partial \tau^2}$, and $\frac{\partial^2 \mathbf{P}'}{\partial S \partial \tau}$. We start with the last one and prove on the way that the apparent tangential velocity is determined by the curvature of (c_τ) and the normal motion field.

Computing the Apparent Tangential Velocity

Using equation (4.9) and the first equation (4.7), we deduce

$$\frac{\partial^2 \mathbf{P}'}{\partial S \partial \tau} = \begin{bmatrix} v_S \mathbf{t} + u \left(v\kappa \mathbf{n} + \frac{\partial \mathbf{v}^a_{\mathbf{m}}}{\partial s} \right) \\ 0 \end{bmatrix} .$$

Let us now evaluate $\frac{\partial \mathbf{v}^a_{\mathbf{m}}}{\partial s}$. From the definition of $\mathbf{v}^a_{\mathbf{m}}$ — equation (4.10) — and the two-dimensional Frenet formulas (2.2), we infer:

$$\frac{\partial \mathbf{v}^a_{\mathbf{m}}}{\partial s} = \left(\frac{\partial \alpha}{\partial s} - \kappa\beta \right) \mathbf{t} + \left(\kappa\alpha + \frac{\partial \beta}{\partial s} \right) \mathbf{n} . \tag{4.17}$$

Thus,

$$\frac{\partial^2 \mathbf{P}'}{\partial S \partial \tau} = \begin{bmatrix} \left(v_S + u \left(\frac{\partial \alpha}{\partial s} - \kappa\beta \right) \right) \mathbf{t} + u \left(\kappa w + \frac{\partial \beta}{\partial s} \right) \mathbf{n} \\ 0 \end{bmatrix} . \tag{4.18}$$

Computing $\frac{\partial^2 \mathbf{P}'}{\partial \tau \partial S}$; from equation (4.8), we deduce

$$\frac{\partial^2 \mathbf{P}'}{\partial \tau \partial S} = \begin{bmatrix} u_\tau \mathbf{t} + u\dot{\mathbf{t}} \\ 0 \end{bmatrix} . \tag{4.19}$$

Using the derivation rule (4.13), we write

$$\dot{\mathbf{t}} = w \frac{\partial \mathbf{t}}{\partial s} + \partial_{\mathbf{n}_\beta} \mathbf{t} = \kappa w \mathbf{n} + \partial_{\mathbf{n}_\beta} \mathbf{t} . \tag{4.20}$$

The vector $\partial_{\mathbf{n}_\beta} \mathbf{t}$ is a derivative of the unit vector $\mathbf{t}$, it is therefore perpendicular to $\mathbf{t}$, thus in the plane defined by $\mathbf{n}$ and τ. We write

$$\partial_{\mathbf{n}_\beta} \mathbf{t} = \rho \mathbf{n} + \eta \tau . \tag{4.21}$$

Therefore, equation (4.20) yields

$$\dot{\mathbf{t}} = (\kappa w + \rho) \mathbf{n} + \eta \tau . \tag{4.22}$$

From Schwartz equality $\frac{\partial^2 \mathbf{P}'}{\partial S \partial \tau} = \frac{\partial^2 \mathbf{P}'}{\partial \tau \partial S}$, and $u_\tau = v_S$. We conclude, by equating equations (4.18) and (4.19) that, if $u \neq 0$, $\eta = 0$, $\frac{\partial \alpha}{\partial s} = \kappa\beta$, and $\rho = \frac{\partial \beta}{\partial s}$. We call ρ the β-curvature of (c_τ) while κ is the space curvature.

We have thus proved the following theorem:

Theorem 4.1 *The tangential apparent motion field α and the β-curvature ρ satisfy:*

$$\frac{\partial \alpha}{\partial s} = \kappa\beta \,, \tag{4.23}$$

$$\rho = \frac{\partial \beta}{\partial s} \,. \tag{4.24}$$

Corrolary 4.1 *The β-derivative of* $\mathbf{t}$ *and* $\mathbf{n}$ *are:*

$$\partial_{\mathbf{n}_\beta} \mathbf{t} = \frac{\partial \beta}{\partial s} \mathbf{n} \,, \tag{4.25}$$

$$\partial_{\mathbf{n}_\beta} \mathbf{n} = -\frac{\partial \beta}{\partial s} \mathbf{t} \,. \tag{4.26}$$

Equation (4.23) is instructive. Indeed, it shows that α, the tangential component of the apparent motion field $\mathbf{v}^a_\mathbf{m}$ is entirely determined up to the addition of a function of time by the normal component of the motion field β and the space curvature κ of (c_τ):

$$\alpha = \int_{s_0}^{s} \kappa(t,\tau)\beta(t,\tau)dt \,. \tag{4.27}$$

Changing the origin of the arclength parameterization from s_0 to s_1 on (c_τ) is equivalent to adding the function $\int_{s_0}^{s_1} \kappa(\gamma,\tau)\beta(\gamma,\tau)d\gamma$ to α, function which is constant on (c_τ). This is the fundamental result of this section. We have proved the following theorem:

Theorem 4.2 *The tangential apparent motion field can be recovered from the normal flow up to the addition of a function of time through equation (4.27).*

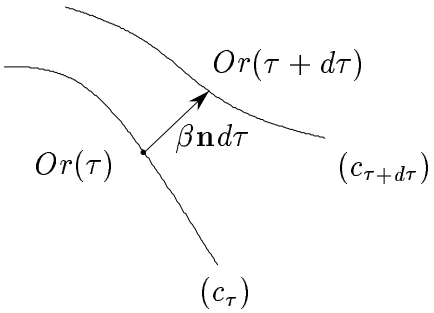

Figure 4.12: The choice of the origin of arclength on (c_τ).

The choice of this function of time is related to the choice of the origin of the arclength parameterization at each time instant. In practice, we want this function of time to be smooth, i.e if $Or(\tau)$ is the origin at time τ, its tangential apparent velocity is 0 by definition (equation (4.27)), and we would like the origin at time $Or(\tau + d\tau)$ to be $Or(\tau) + \beta \mathbf{n} d\tau$ (see figure 4.12). Mathematically, this means that if $\mathbf{P}_{or}$ is the corresponding point on (Σ), it follows a trajectory described by the following differential equation[2]:

$$\frac{d\mathbf{P}_{or}}{d\tau} = \mathbf{n}_\beta \ . \tag{4.28}$$

We know from the theory of differential equations that if $\mathbf{n}_\beta$ is smooth enough, then equation (4.28) has a unique solution for the initial condition $\mathbf{P}_{or}(0) = \mathbf{P}_0$ in a neighborhood of $\mathbf{P}_0$. Thus, since $\mathbf{n}_\beta(\tau)$ is known at each time instant, if we choose the origin as $\mathbf{P}_0$ at time 0, we can compute what the origin is at time τ, and compute α from equation (4.27).

We now prove an interesting relationship between the β- and space curvatures of (c_τ).

Proposition 4.4 *The β- and space curvatures of (c_τ) satisfy*

$$\frac{\partial \rho}{\partial s} = \partial_{\mathbf{n}_\beta} \kappa - \kappa^2 \beta \ ,$$

[2] Or equivalently, the point Or in the image follows a trajectory defined by: $\frac{dOr}{d\tau} = \beta \mathbf{n}$.

which can be rewritten as:

$$\partial_{\mathbf{n}_\beta}\kappa = \frac{\partial^2\beta}{\partial s^2} + \kappa^2\beta \ . \tag{4.29}$$

Proof: We compute $\frac{\partial^2\mathbf{t}}{\partial s\partial\tau}$ and $\frac{\partial^2\mathbf{t}}{\partial\tau\partial s}$ and write that they are equal. Using the derivation rule (4.12), we can write:

$$\frac{\partial^2\mathbf{t}}{\partial s\partial\tau} = \frac{\partial\left(\alpha\frac{\partial\mathbf{t}}{\partial s} + \partial_{\mathbf{n}_\beta}\mathbf{t}\right)}{\partial s} = \frac{\partial((\kappa\alpha+\rho)\mathbf{n})}{\partial s} \ .$$

Thus:

$$\frac{\partial^2\mathbf{t}}{\partial s\partial\tau} = \left(\frac{\partial\kappa}{\partial s}\alpha + \kappa\frac{\partial\alpha}{\partial s} + \frac{\partial\rho}{\partial s}\right)\mathbf{n} - \kappa(\kappa\alpha+\rho)\mathbf{t} \ .$$

On the other hand:

$$\frac{\partial^2\mathbf{t}}{\partial\tau\partial s} = \frac{\partial(\kappa\mathbf{n})}{\partial\tau} = \alpha\frac{\partial(\kappa\mathbf{n})}{\partial s} + \partial_{\mathbf{n}_\beta}(\kappa\mathbf{n}) \ ,$$

thus,

$$\frac{\partial^2\mathbf{t}}{\partial\tau\partial s} = \left(\alpha\frac{\partial\kappa}{\partial s} + \partial_{\mathbf{n}_\beta}\kappa\right)\mathbf{n} - \kappa(\kappa\alpha+\rho)\mathbf{t} \ .$$

From where the announced results follow. □

Computing the Coefficients of the Second Fundamental Form

Let us now evaluate $\frac{\partial^2\mathbf{P}'}{\partial S^2}$ and $\frac{\partial^2\mathbf{P}'}{\partial\tau^2}$; from equation (4.8) and the first equation of (4.7) we derive

$$\mathbf{P}'_{S^2} = u^2\mathbf{P}_{s^2} + u_S\mathbf{P}_s = \begin{bmatrix} u^2\kappa\mathbf{n} + u_S\mathbf{t} \\ 0 \end{bmatrix} \ ,$$

and from equation (4.9):

$$\mathbf{P}'_{\tau^2} = \begin{bmatrix} v_\tau \mathbf{t} + v\dot{\mathbf{t}} + \dot{\mathbf{v}}^a_m \\ 0 \end{bmatrix} .$$

We have seen previously that $\dot{\mathbf{t}} = (\kappa w + \rho)\mathbf{n}$ (equation (4.22)); now, using again the derivation rule (4.13), we obtain

$$\dot{\mathbf{v}}^a_{\mathbf{m}} = w \frac{\partial \mathbf{v}^a_{\mathbf{m}}}{\partial s} + \partial_{\mathbf{n}_\beta} \mathbf{v}^a_{\mathbf{m}} .$$

From equations (4.17) and (4.23), we know that $\frac{\partial \mathbf{v}^a_{\mathbf{m}}}{\partial s} = (\kappa\alpha + \rho)\mathbf{n}$, so evaluating $\partial_{\mathbf{n}_\beta} \mathbf{v}^a_{\mathbf{m}}$ from (4.10),(4.21) and (4.26) yields

$$\partial_{\mathbf{n}_\beta} \mathbf{v}^a_{\mathbf{m}} = (\partial_{\mathbf{n}_\beta} \alpha - \beta\rho)\mathbf{t} + (\alpha\rho + \partial_{\mathbf{n}_\beta} \beta)\mathbf{n} .$$

Finally,

$$\mathbf{P}'_{\tau^2} = \begin{bmatrix} (v_\tau + \partial_{\mathbf{n}_\beta} \alpha - \beta\rho)\mathbf{t} + (\kappa w^2 + 2w\rho + \partial_{\mathbf{n}_\beta} \beta)\mathbf{n} \\ 0 \end{bmatrix} .$$

We can now compute the coefficients of the second fundamental form; after some algebra, and using equation (4.24), we obtain:

Proposition 4.5 *The coefficients of the second fundamental form in the basis* $(\mathbf{P}'_S, \mathbf{P}'_\tau)$ *of* $\mathbf{T}_P$ *are given by:*

$$L = \frac{\kappa u^2}{\sqrt{1+\beta^2}} , \quad M = \frac{(\kappa w + \frac{\partial \beta}{\partial s})u}{\sqrt{1+\beta^2}} , \quad N = \frac{\kappa w^2 + 2w\frac{\partial \beta}{\partial s} + \partial_{\mathbf{n}_\beta} \beta}{\sqrt{1+\beta^2}} . \tag{4.30}$$

We can also compute those coefficients in the basis $(\mathbf{t}_0, \mathbf{n}_\beta)$:

Proposition 4.6 *The coefficients of the second fundamental form in the basis* $(\mathbf{t}_0, \mathbf{n}_\beta)$ *are given by:*

$$L' = \frac{\kappa}{\sqrt{1+\beta^2}} , \quad M' = \frac{\frac{\partial \beta}{\partial s}}{\sqrt{1+\beta^2}} , \quad N' = \frac{\partial_{\mathbf{n}_\beta} \beta}{\sqrt{1+\beta^2}} .$$

Proof:

Let us denote ψ the linear mapping $\mathbf{T}_P \rightarrow \mathbf{T}_P$ such that $\mathbf{\Phi}_2\mathbf{x} = \psi\mathbf{x}\cdot\mathbf{x}$ for all $\mathbf{x}$ of $\mathbf{T}_P$. Since we have

$$\mathbf{t}_0 = \tfrac{1}{u}\mathbf{P}'_S \ , \quad \mathbf{n}_\beta = -\tfrac{w}{u}\mathbf{P}'_S + \mathbf{P}'_\tau \ ,$$

we can write

$L' = \mathbf{\Phi}_2\mathbf{t}_0 = \frac{1}{u^2}\mathbf{\Phi}_2\mathbf{P}'_S = \frac{L}{u^2}$, $N' = \mathbf{\Phi}_2\mathbf{n}_\beta = \frac{w^2}{u^2}L - 2\frac{w}{u}M + N = \frac{\partial_{\mathbf{n}_\beta}\beta}{\sqrt{1+\beta^2}}$, and $M' = \psi\mathbf{t}_0\cdot\mathbf{n}_\beta = -\frac{w}{u^2}L + \frac{1}{u}\psi\mathbf{P}'_S\cdot\mathbf{P}'_\tau = -\frac{w}{u^2}L + \frac{1}{u}M = \frac{\frac{\partial\beta}{\partial s}}{\sqrt{1+\beta^2}}$.
□

This proposition shows that neither u nor v can be recovered from the first and second fundamental forms of (Σ). Indeed, we have seen in section 2.4 that the quantities of interest (i.e. invariant with respect to changes of the parameterization of (Σ)) are the principal directions and curvatures which depend only on the coefficients E',F',G' and L',M',N'. Since none of these quantities depend upon u and v this is also true of the principal directions and curvatures. This means that if we observe (Σ) and compute its differential invariants, we will not be able to recover u and v and therefore, we will not be able to recover the real tangential motion field $w = \alpha + v$.

4.4.3 The Mainardi-Codazzi Equations

The reader may wonder whether we have completely characterized the spatio-temporal surface (Σ) and if it is not possible to find other relations that may yield more information than what we have found so far. The answer to this is no, thanks to the Bonnet theorem. We show that by combining equations (2.9)-(2.11) with the expressions (4.14) and (4.30) that the Gauss and Mainardi-Codazzi equations (2.9), (2.10), (2.11) for (Σ) imply equations (4.29) and (4.23). With the help of a computer algebra system, it can be shown that the Gauss and Mainardi-Codazzi equations are given by:

$$\frac{wu\beta}{(1+\beta^2)^{\frac{3}{2}}}\left(\kappa^2\beta + \frac{\partial^2\beta}{\partial s^2} - \partial_{\mathbf{n}_\beta}\kappa\right) = 0 \ ,$$

$$\frac{u}{(1+\beta^2)^{\frac{3}{2}}}\left(\kappa(1+\beta^2)-\partial_{\mathbf{n}_\beta}\beta\right)(u_\tau - w_S + \kappa u\beta) = 0 \ ,$$
$$\frac{u}{(1+\beta^2)^{\frac{1}{2}}}\left(\frac{\partial\alpha}{\partial s}\frac{\partial\beta}{\partial s} - \partial_{\mathbf{n}_\beta}\left(\frac{\partial\beta}{\partial s}\right) + \frac{\partial\left(\partial_{\mathbf{n}_\beta}\beta\right)}{\partial s}\right) = 0 \ .$$

The first equation is equivalent to equation (4.29) in the general case where w, u and β are different of 0. Using the facts that $w = v + \alpha$, $v_S = u_\tau$, and $\alpha_S = u\frac{\partial\alpha}{\partial s}$, we can write the second equation as:

$$\frac{u^2}{(1+\beta^2)^{\frac{3}{2}}}(\kappa(1+\beta^2)-\partial_{\mathbf{n}_\beta}\beta)\left(\frac{\partial\alpha}{\partial s}-\kappa\beta\right) = 0$$

This is equivalent to (4.23) if $\partial_{\mathbf{n}_\beta}\beta \neq \kappa(1+\beta^2)$ which is, in general, true. We show it later in the simple case of the retinal rigid motion (see appendix B).

It is easy to recognise in the term $\frac{\partial\left(\partial_{\mathbf{n}_\beta}\beta\right)}{\partial s} - \partial_{\mathbf{n}_\beta}\left(\frac{\partial\beta}{\partial s}\right)$, which appears in the second factor of the third equation, the Lie bracket $L_{[\mathbf{n}_\beta,\mathbf{t}_0]}$ applied to β. Let us thus compute this quantity.

Equation (2.12) given in section 2.4.3 shows how to compute the coordinates of $[\mathbf{n}_\beta,\mathbf{t}_0]$ from those of $\mathbf{t}_0$ and $\mathbf{n}_\beta$. Moreover, we have seen in section 2.4.3 how to compute the Lie derivative in a given direction of $\mathbf{T}_P$ given the coordinates of that direction in the frame $(\mathbf{P}'_S,\mathbf{P}'_\tau)$. Thus, the coordinates of $\mathbf{t}_0$ and $\mathbf{n}_\beta$ in this frame being given by equation (4.15), we have:

$$\begin{aligned}[\mathbf{n}_\beta,\mathbf{t}_0] &= \left(-\frac{w}{u}\frac{\partial\left(\frac{1}{u}\right)}{\partial S} - \frac{1}{u}\frac{\partial\left(-\frac{w}{u}\right)}{\partial S} + \frac{\partial\left(\frac{1}{u}\right)}{\partial\tau}\right)\mathbf{P}'_S + 0\mathbf{P}'_\tau \ , \\ &= \frac{w_S - u_\tau}{u^2}\mathbf{P}'_S = \frac{\alpha_S}{u^2}\mathbf{P}'_S \ ,\end{aligned}$$

since $w = \alpha + v$ and $v_S = u_\tau$. Consequently, the Lie bracket $L_{[\mathbf{n}_\beta,\mathbf{t}_0]}$ is given by

$$L_{[\mathbf{n}_\beta,\mathbf{t}_0]} = \frac{\alpha_S}{u^2}\frac{\partial}{\partial S} = \frac{\partial\alpha}{\partial s}\frac{\partial}{\partial s} = \frac{\partial\alpha}{\partial s}L_{\mathbf{t}_0} \ . \tag{4.31}$$

Applying this result to β yields

$$L_{[\mathbf{n}_\beta,\mathbf{t}_0]}\beta = \partial_{\mathbf{n}_\beta}(\frac{\partial\beta}{\partial s}) - \frac{\partial(\partial_{\mathbf{n}_\beta}\beta)}{\partial s} = \frac{\partial\alpha}{\partial s}\frac{\partial\beta}{\partial s} \ ,$$

which proves that the third equation is always true.

From the Bonnet theorem, the two equations (4.23) and (4.29), together with those giving the coefficients of the first and second fundamental forms (equations (4.14) and (4.30)), completely characterize (Σ), up to a rigid motion. But since (4.23) and (4.29) do not depend upon u and v, they cannot help us to recover w from (Σ).

Theorem 4.3 *The tangential real motion field cannot be recovered from the spatio-temporal surface.*

4.5 Joint Experiment between Computer Vision and Neurophysiology

There are some results in neurophysiology that tends to show biological vision systems do overcome the aperture problem. From our previous study, it is hypothesized that the obtained motion field is the apparent one: actually, it is possible to recover the real motion field if some more assumptions are made about the 3D motion like rigidity (see part III). However, in such a case, the tangential component of this field is deduced from the 3D motion and since the area of the brain (the middle temporal area MT) occurs at a quite early stage of the "processing" of the retinal data, this is fairly improbable. Designing and realizing an experiment to test this hypothesis is the topic of a joint experiment between the group of neurophysiology directed by Prof. Orban (KULVNP) and INRIA in the framework of the European project INSIGHT II.

Actually, the situation is slightly more complex. There is a good evidence that in the monkey MT area, an optical flow map is computed. Cells in this area are both direction and velocity tuned [MVE83, LRXO90]. Moreover, there is a strong evidence that these MT cells only react to local situations. Consequently, it seems that they are not using any global data to deliver an output. Furthermore, the design of a neurophysiological experiment is a very difficult task: the visual stimulus must

provide only those cues which the tested cell is known to be sensitive to. From this point of view, the experiment showing that the biological vision systems overcome the aperture problem is questionable since it used two intersecting nets of straight lines as the visual stimulus and since such points may provide a strong cue about the real field. We have thus tried to design a new experiment in order to clarify the two following question:

- The first question to answer is whether neurons of this MT area are measuring the normal flow or a full flow along an edge.
- If the answer to the previous question is that MT cell measure a full flow then it would be interesting to check if this flow is the real or the apparent velocity field.

To achieve such a goal, we have used some of the tools developed for generating synthetic image stimuli sequences of rigid 3D curves (see part II).

4.5.1 Principle of the Experiment

The basic idea of the proposed experiment is to take a neuron for which field of view and the direction preference are known, and to stimulate it twice. The image sequence is a film of a translating pattern and the two different stimuli are obtained from each other by a symmetry with respect to the direction of translation. This operation has the advantage of preserving the direction of the real flow whereas the normal flow is changed since the orientation of the edge is changed by the operation. If the neuron reacts the same way with the two stimuli then it is sensitive to the real motion field, if it does not then if measures either the normal component of this field or something else (the apparent motion field ?). The same kind of experimental protocol can be used to discriminate between the apparent and real motion field although it is more complex to design. The main difficulty encountered when generating these image sequences is to choose the curve in such a way that the range of the orientations along the viewed part of the curve is as small as possible (in order to be able to say that there is one main orientation of the curve in the receptive field) while preserving the ability of the visual system to find a unique motion. Furthermore, there must be as few other cues as possible in the stimulus. Of course, these two constraint are somewhat incompatible and some optimal choice must be made.

4.5.2 Examples of Stimuli

We generated many different sequences. Each of these sequences shows a set of rigid planar curves moving together with the same motion. To obtain these curves we have taken the model defined by the equation

$$y = x^3 + \mu x + translation\ .$$

The nice property of this model is that it have one clear main direction in the image. Moreover, the parameter μ controls the value of the derivative at the inflexion point[3] of the curve and the parameter *translation* allows to translate arbitrarily the curve along the y axis: this allows some flexibility in the design of the image sequences and allows the tuning of the parameters to optimize the criteria described in the previous section. To obtain space curves, we extended the (x, y) plane with a z axis defined as the normal to the plane and added a rotation with an angle α around the x axis. Thus the motion observed is the translation in the tilted plane of a rigid planar cubic defined in that plane. This generic model of curve is then projected onto a camera whose optical center is on the z axis and whose retina is parallel to the (x, y) plane. We have the mathematical equations of this generic projection, it is thus easy to obtain some statistics or probability measures of the repartition of the normals.

The image is defined as the set of such rigid curves obtained by varying *translation* parameter for each curve (this same parameter is again used for all the curves globally to generate the motion). We thus have a computer program that takes the name of the sequence, several time sampling parameters, the α and μ parameters and a file giving the translation values from one curve to an another and the grey level value between two curve and produces a 512×512 image sequence.

We have generated many such sequences giving the values of 0.0, 0.1, 0.2 radians for the α parameter and the values of 0.0, 0.25 and 0.5 for the μ parameter. We have limited ourselves to these values because higher angles give rise to very strong perspective effects, and because the visual effect obtained with higher values of the μ parameter is more and more non-rigid (although the model is strictly rigid) according to what have been observed by Nakayama.

[3] The presence of the inflexion point seems not to be a cue problem since it seems that the visual system does not track this kind of points.

Figure 4.13 show one image excerpted from each sequence. Figure 4.14 is excerpted from a sequence for which the μ parameter is 0.25 with a tilt angle of $\pi/6$ radians and shows the strong perspective distortion that we obtain. Figure 4.15 show the next generation of stimuli we have developed. In these, we have added a surrounding circle to obtain a circular field of view and to minimize the compression effects that can be seen on the borders of the images shown in figure 4.13 (these constitute an undesirable cue that hints the visual system toward the actual structure of the scene). At this stage, it seems that using these stimuli would need a psychophysical experiment to assert that the global percept of the visual system is that of a rigid motion. Figure 4.16 shows the distribution of the orientation along the curves of figure 4.15.

It is not yet clear how useful will be these stimuli. Some more experiments are needed to be able to decide if it is possible to define a rigorous experimental protocol based on these. However, the task of designing them has already raised many interesting questions and has pinpointed a lot of possible cues that the biological visual systems might use for motion analysis. This is interesting both for computer vision and for the study of biological visual systems.

4.6 Conclusion

There are three main consequences that we can draw from the analysis made in this chapter. Under the weak assumption of *isometric* motion:

1. The normal motion field β can be recovered from the normal to the spatio-temporal surface (proposition 4.3),
2. the tangential apparent motion field can be recovered from the normal motion field through equation (4.27), up to the addition of a function of time, and we have seen how to eliminate this problem,
3. the tangential real motion field cannot be recovered from the spatio-temporal surface.

Therefore, the full real motion field is not computable from the observation of the image of a moving curve under the isometric assumption. This can be considered as a new statement of the so-called *aperture* problem. In order to solve it we *must*

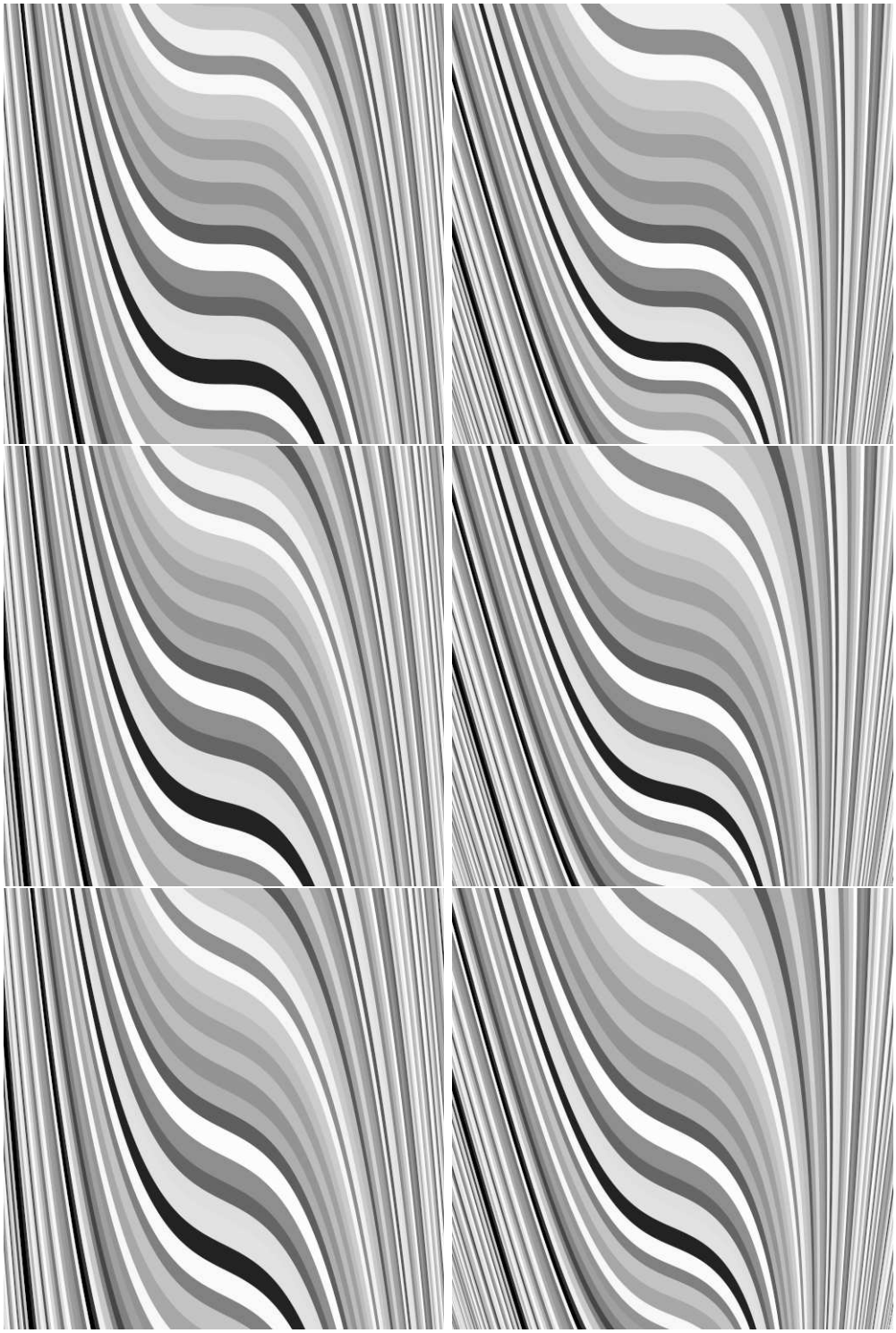

Figure 4.13: Images excerpted from the generated sequences. The left images are fronto-parallel view of the curve net whereas the right ones are obtained with a plane tilted with an angle of 0.2 radians. The rows correspond respectively to derivative values at the inflexion point of 0.0, 0.25 and 0.5.

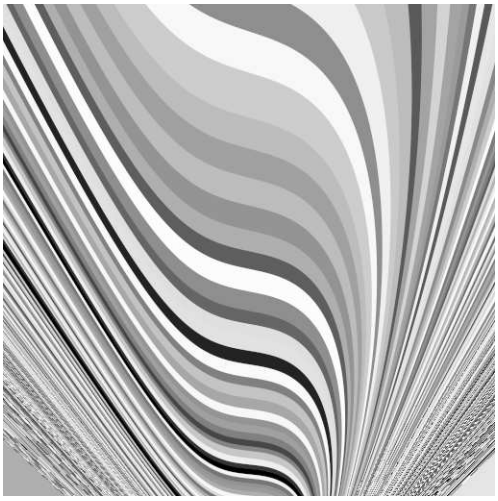

Figure 4.14: An image excerpted from the sequence obtained using a tilt angle of 0.52 (approximatively $\pi/6$) radians and a derivative value of 0.25 at the inflexion point.

add more hypothesis, for example that the 3D motion is rigid. This is what will be done in part III. Note that it is *not* what previous authors have done [HS81, Nag83, Hil83b, Gon89, Bou89]. We suspect that those authors actually compute the apparent motion field which, as we have shown it on an example, can be quite different from the real one. In particular, we show in appendix C that it is the case for the algorithm described in [Bou89]. Furthermore, stimuli have been defined that might be used to understand better how biological systems handle this problem at a low-level (if they handle it at all). Defining such stimuli has already raised a lot of questions about what cues are actually used by those systems. It is hoped that they will be used in a near future to understand better the functioning of biological system. The consequences of this study will certainly be very important for computer vision as well.

We show in chapter 7 that if we assume a 3D rigid motion then the problem is, in general and in theory, solvable but that there is no need to compute the full real motion field.

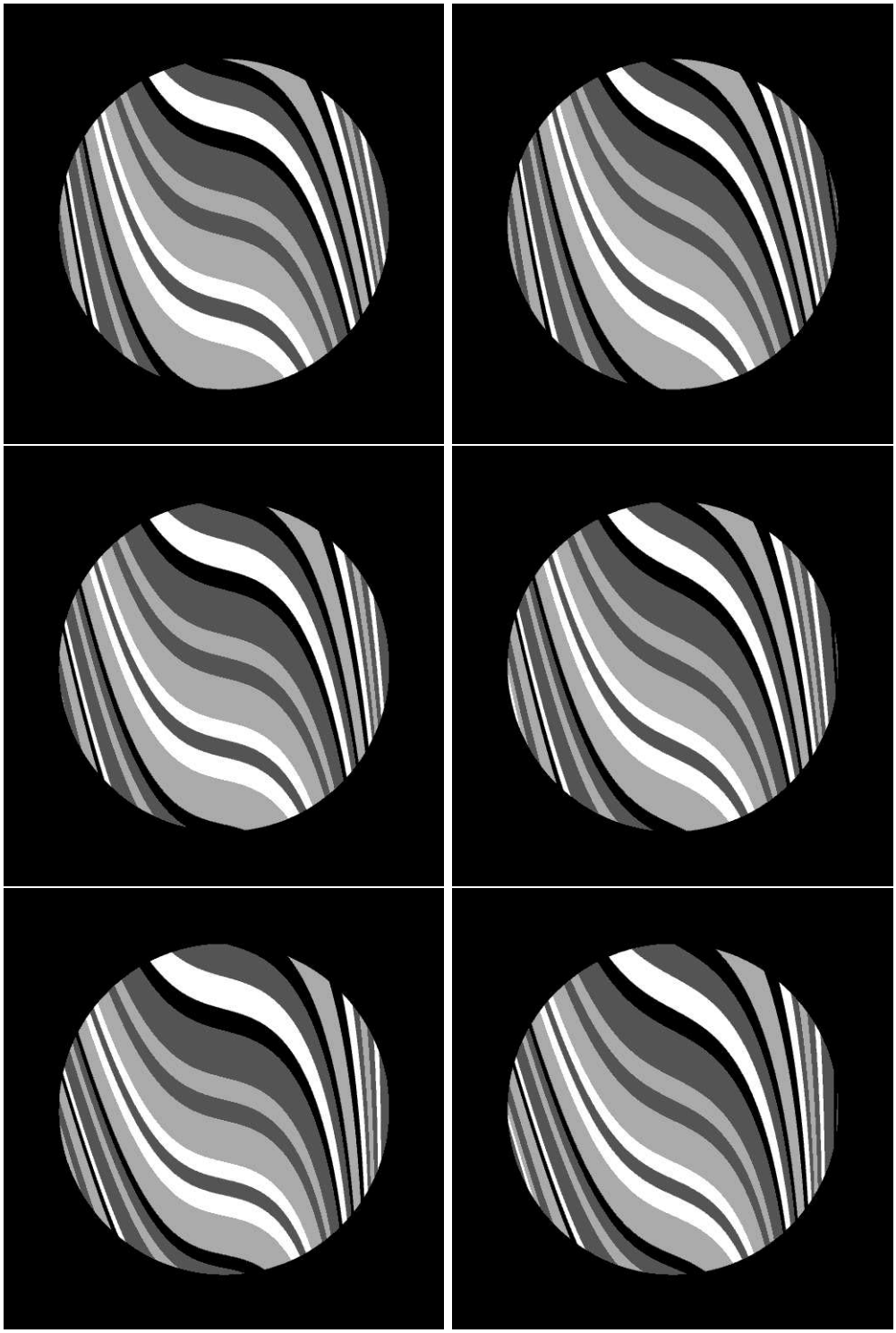

Figure 4.15: Images excerpted from the generated sequences. The left (resp. right) images correspond to a derivative value at the inflexion point of 0.25 (resp. 0.5). The rows correspond respectively to tilt angles of 0.0, 0.1 and 0.2 radians. A circular mask centered at point (256,256) and of radius 192 has been added.

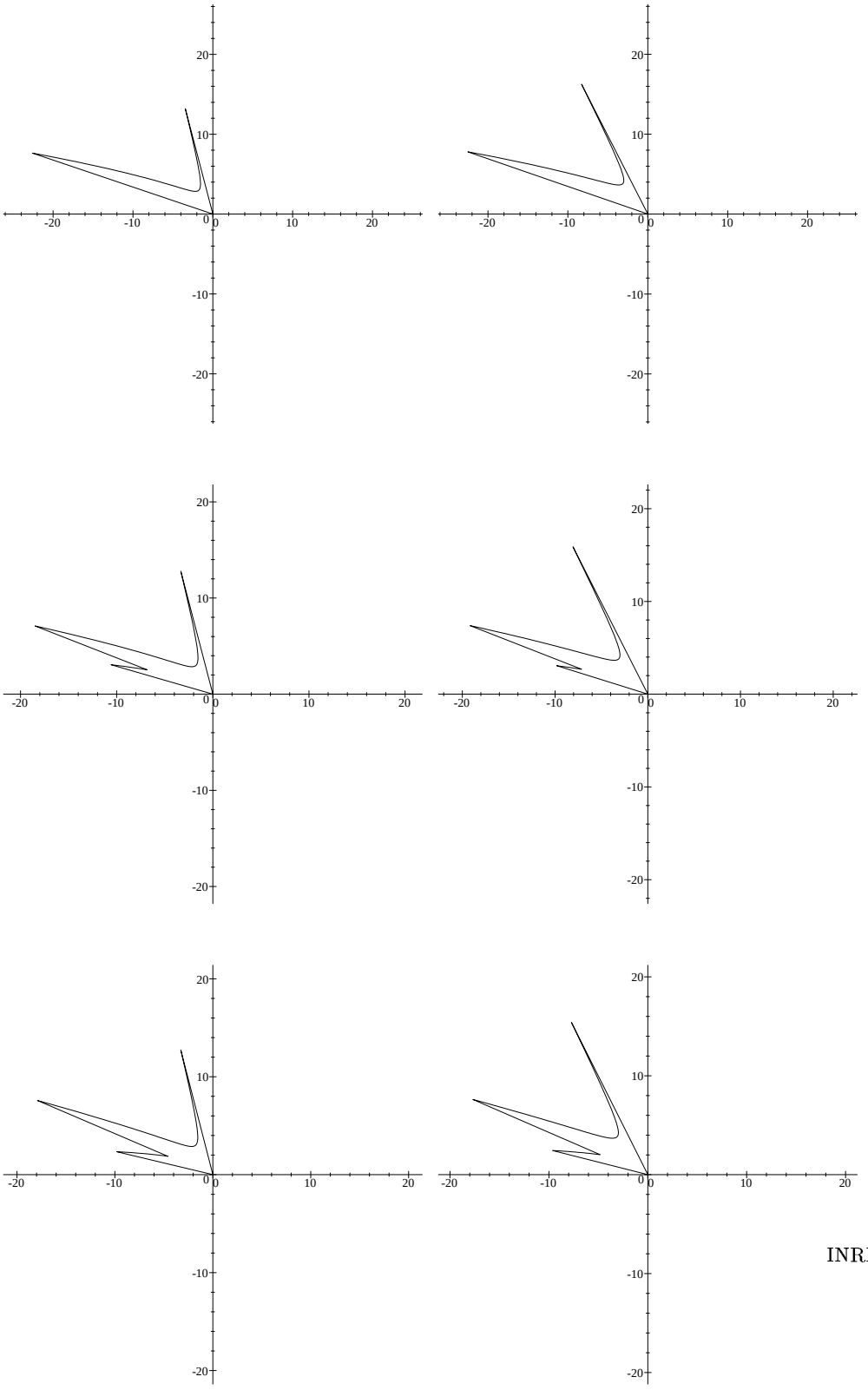

Figure 4.16: The orientation distributions visualized as polar plots. Each plot corresponds to the visible part of a curve in an image. The plots are in correspondence with the images of figure 4.15.

Part II

Derivative Computation

As presented in the introduction, the main goal of this work is the estimation of the motion of a rigid 3D smooth curve. We consider a monocular image sequence of it, and use a pinhole camera model[4]. In the next part, we will see that the measures needed to achieve such a computation are the quantities: $\mathbf{m}$, $\mathbf{n}$, κ, β, $\frac{\partial \beta}{\partial s}$, $\partial_{\mathbf{n}_\beta}\beta$ where we use the notation introduced in chapter 4. None of these are provided directly by the CCD sensors. However, as sketched briefly in chapter 3, many reliable methods have been proposed to extract edges robustly from images [MH80, Can86, Der87]. Applying one of these methods to our images immediately provides for the first of these quantities, $\mathbf{m}$. The remaining ones are derivatives up to the second order in space and time defined only along smooth edges. As such, they are more difficult to obtain from the intensity images — it is well-known that the bigger the order of derivation, the more noisy the results, and in practice, second order derivatives are seldom used in the algorithms of computer vision because they are troublesome to compute accurately. The purpose of this part is to show that computing these quantities is not an unrealistic challenge and that quite good quantitative values can be obtained provided that the curve is sampled sufficiently often.

Basically, the tools developped in this part are completely general and can be used for every differential quantity, but both the methods and the results provided in this part can be used for many other edge based computer vision algorithms that also involve their quantitative differential properties [AB86, WKPS87, Sub88, RF91, VMPO92, KTZ92]. Since derivatives hold a very important information about the contour deformation (as we will see it for our case in the part), it is likely that the techniques discussed in these works, as well as other techniques based on differential properties, will play a more and more important role in computer vision. The high resolution devices that are emerging nowadays will certainly ease the generalization of these techniques to implementation within many of the practical situations.

Many methods have been proposed in the litterature to compute such derivatives. They can be amalgamated in two main categories:

Global methods: this kind of method first computes a smooth global model that fits to the data (image intensities or edge points), and then uses it to compute all of the desired derivatives. The efficiency of this approach in terms of preciseness is directly related to the number of parameters involved in the model.

[4]The importance of the rigidity assumption will appear clearly in the next part. For now, let us just say that its main advantage is to provide for a way to globally describe the motion of the curve by a finite set of parameters.

These are closely related to the flexibility of the model. Generally, this kind of method is very efficient with a very constrained model (compatible with the data of course), such as for example a straight line or a conic. In such cases, the computation of a derivative at one point of the edge takes advantage of all the data points that belong to the edge; this gives rise to very precise derivatives (at least up to order 2). Unfortunately, an a priori model is seldom known. The technique that is used consequently is to fit a very flexible model such as a spline. Yet, this leads to a global model but with a much larger number of parameters. The main problem in this case becomes the control of the balance between the flexibility of the model and the smoothing that we want to occur in order to obtain good derivatives. Again, derivatives can be computed from the global model [BD92, GA92], but they usually tend to be less accurate than in the previous case. Actually, splines fitting can be considered as lying inbetween the pure global method (line or conic fitting for example) and pure local methods as described hereafter.

Local methods: When no model is known about the observed curve, another method used to compute derivatives is to look at the edge only locally at a point. Namely, this means looking in a neighborhood of each edge point, and computing the derivatives given the data (here again data can be either intensities or edge points) that are in this neighborhood. Here there is no global information that is conveyed from one point (outside of the neighborhood) to another. Usually these methods give results that are quite noisy, but we show here that, provided that the curve is well enough sampled, it is possible to recover quite good derivative values using them.

In what follows, focus is made on local methods. There are three main reasons for this choice:

- First, the method used to recover the 3D motion of a rigid curve involves an equation defined at each point of the observed edge. It is thus more natural to work locally around each point than to use a global method.
- The second reason is that since we do not generally have a constraining model of the observed curve, a very flexible global model would have had been to have used. As mentioned above, these global models are not very different from local ones. For example, splines can be considered as local models pasted

together with some continuity conditions. By using pure local models, we get rid of these extra conditions that are not needed in our application thus obtaining more local freedom to control the approximation.

- The third reason is more philosophic: derivatives are inherently local measures of a curve behaviour thus it seems more natural to use local methods.

In the remainder of this part, we suppose the following situation: suppose that we have a continuous sequence of images in which there is a smooth curve tracked over time. Smoothness and continuity here mean that the image curve varies slowly with respect to space and time: this ensures that all the image curves are smooth and vary slowly. As a consequence, it allows us to consider high frequency variations as noise. Given this set of images, by using standard edge detectors it is possible to obtain a set of points that belong to the spatio-temporal surface swept out by the curve. Although we focus on algorithms that work on the set of the extracted points, some of the methods we present (mostly for comparison purposes) rely directly on image intensities. In the remainder of this part, unless otherwise specified, the word edge must be taken as "generalized edge". This means that an edge is not necessarily a curve, it can as well be a spatio-temporal surface which is nothing else but the spatio-temporal equivalent of an edge.

In the next chapters, several methods for computing derivatives are reviewed and compared. The first chapter describes the general outline of the method as well as the experimental protocol that was used to test the quality of our estimators. The chapter which follows shows how this general method is applied to compute the quantities we are interested in, gives the obtained results, and discusses both the merit of the different methods and the importance of some stages of the computation. A significant part of the discussion is devoted to the accuracy of the results for orientation and curvature measures. This is because much attention has been paid to these two quantities in the literature. Some conclusions about what makes derivatives difficult to compute from images are drawn from these examples.

Chapter 5

Sketching the Method

This chapter discusses the general choices that have been made in the design of our derivative estimation method. For the reasons explained in the introduction of this part, we have resolutely focussed on a method that does not appeal to a global representation of the observed curve. Basically, the looked-for method must fulfill the following goals:

- The values of the computed derivatives must be quantitatively good. As we are planning to use these values in some equations, it is very important to have accurate values and not only good qualitative estimates. This differs from some of the computer vision algorithms that, for robustness reasons, rely only on some global characteristics of the differential quantities such like the extrema or the zero crossings.
- The method must be totally general. This means that, apart from the smoothness needed for defining the differentiation operation, the algorithm must not depend on any geometric constraint imposed to the observed curves. In general, our input is an ordered set of points and the algorithm must give a value for the derivative for each point of the curve (except may be for those that are on the sides of the edge).

- Finally, but this is very important, we prefer methods that work with non uniformly sampled data points. This constraint arises from the fact that the data points corresponding to a curve extracted on a regularly sampled image are *not* regular samples on the curve! This phenomenon is amplified by the fact that our parameters are defined only in normalized image coordinates: the affine transform that relates image and normalized coordinates generally involves two different scale factors for the two axes spanning the retina so that usually any uniform sampling is destroyed by this transform.

These constraints are quite important. On another side, in the tradeoff between speed and accuracy, we have clearly opted for accuracy so that the speed of the algorithm is not that important (for example, no attempt was made to make the program work in real time). Furthermore, in order to reach the fixed goals, we assume that the observed curve is sampled well enough so that the looked-for information is present in the data points. Throughout this work, there is the underlying assumption that the image data are just a discretized view of some continuous physical situation. When we say that we deal with smooth curves, the word "smooth" clearly applies to the latter situation[1].

This chapter is organized in four sections. First a short review of local methods is made. Then, the general description of the one we have chosen to follow is given. The details of the fitting procedure used with this method are given in the third section. Finally, the last section deals with the experimental protocol we have used to validate the estimators.

5.1 Local Methods

Even when focusing on local methods, there are many different ways to compute derivatives. Basically, we can split them in two classes: image based methods and point based ones. In this section, we briefly describe these two classes and justify the choice we have made. Although most of the matter presented in this section can be applied to the computation of any differential quantity, the example of curvature is principally taken. This is because, contrarily to the other quantities (except for the

[1] It is much more difficult to deal with discrete data than with continuous ones. Actually, as soon as dicretization occurs, there is a non-obvious relation between the scale of the details we want to observe and the sampling of the data. The same phenomenon appears with derivative computation.

orientation and the normal flow but these are only first order derivatives), curvature have received a lot of attention in the literature.

5.1.1 Image based methods

These are methods that compute the derivatives directly from image intensities: for these methods, the image intensities are considered as a function $I(x, y, \tau)$. For each edge extraction algorithm, it is possible to characterizes edges by some differential properties of this function. This constraint can always be stated as a function f relating I and its derivatives (usually up to the second order): $f(I, I_x, I_y, I_\tau, I_{xx}, I_{xy}, I_{yy}) = 0$. Doing so defines an edge as an implicit function in the unknowns x, y and τ, and, as shown for example for curvature in section 2.4, it is possible to express any derivative measured on the spatio-temporal surface in terms of f and of its derivatives with respect to x, y and τ (the formulae for all the quantities we want to measure are given in chapter 10). Practically, this yields expressions depending on the image derivatives. These can be computed using standard filtering techniques and directly combined to obtain the desired quantity at each edge pixel.

The simplest example of this technique is also the most frequently used one: it consist in assimilating edges with isointensity curves. Then, intensity derivatives and edge derivatives match exactly along the edge and the formulae of chapter 10 for implicit curves works directly with image derivatives. The computational scheme is thus:

1. Compute the image intensity derivatives.
2. Combine all these derivatives with the formulae of chapter 10 to obtain the desired quantity.

This technique has been used extensively in computer vision to estimate many kinds of differential properties of edges (see [HS81, Nag92, KKvD93] for example). In the next chapter, curvatures obtained with this method and the one described hereafter will be compared.

More details about the generalization of this scheme to different edge criteria are given in [FP93]. A full description of the method with some results can be found in [SFP94]. Notice however that the higher the derivative order used in the function

f, the higher will be the derivation order needed to express the result. Taking curvature as example, second order derivatives with respect to the space coordinates are sufficient when assimilating edges with isointensity curves whereas fourth order spatial derivatives are needed with the models used in the Canny-Deriche or the Marr-Hildreth edge detectors.

A very nice refinement of this kind of method is to compute the intensity derivatives at several different scales, and then to look for the desired quantities in this "scale space". Unfortunately, until now, there is no well established method to decide at which scale we have to search for the desired value. This is all the more annoying as, in general, different parts of a curve must be looked at different scales (we will try to demonstrate this later). Notice, however, that these last years have seen the emergence of a neat axiomatisation of the ideas of scale space: see for example [AGLM92, FtKV92, KKvD93]. A generalization of these ideas to image sequences is presented in [Gui94].

One of the main advantages of this kind of method is that it is quite easy to implement once the image derivatives have been obtained. However, if accuracy is needed, one has to be very careful of how to select the pixel values at which to pick the derivative values (actually, sometimes the signal around a pixel is varying so fast that interpolating between two pixel given a subpixelic edge position is a necessity). Apart from this one, there are two main drawbacks with this kind of method:

- They are computationally intensive especially if the scale space paradigm is used.
- Standard filters that compute derivatives (usually for edge detection purposes) are not always normalized. This means that they compute derivatives only up to a scale factor, so if one want to combine together different types of derivatives, this normalization factor must be recovered. Notice also the problems arising from photometric effects (see chapter 3) and especially those implicated by automatic gain control if it is used during the sequence acquisition.

Finally, a slightly different approach is described in [VF94]. In this approach, the image derivatives are not computed directly from the image intensities but merely from a model (a polynomial model for example) fitted on the intensity. The underlying assumption is that, locally, edges are isointensity curves.

5.1.2 Point based methods

Point based methods use the edge point coordinates obtained after edge detection. Many different variants of these can be designed. For example, for the curvature computation problem, Worring and Smeulders [WS93] claim that there are essentially three equivalent formulations of curvatures (methods based on orientation, methods based on paths and those based on the osculating circle) which give rise to five different implementations (basically the article proposes three different ways of estimating the orientation). Theoretically, all these methods should give the same results but the quality of the practical results show errors ranging from 1% to 1000%! Among these methods, those based on the orientation and on the osculating circle are very specific to curvatures and are quite difficult to generalize to other quantities. Moreover, path and orientation based methods can be reformulated in such a way that they appear to be very close.

- Path based methods [MM86, Low88, MM92] are based on the recovery from image data of a local parametric representation $(x(s), y(s))$ of the edge. Once this representation has been computed, the standard formula for curvature of parametric curves (see chapter 2.4) is used. In the following discussion, we use a slightly more general definition of this kind of method in which we allow ourselves to start with any set of quantities defined on the curve: x and y are two such quantities used in the previous description, but now any other quantity defined along the curve (such as orientation, curvature or normal flow) can be used.

- Orientation based methods are based on the following formula for curvature:

$$\kappa = \frac{d\theta}{ds} ,$$

  where θ is the local orientation of the curve and s is the arclength. A typical example of such a method is given in [AB86]. Provided that care is taken, this kind of methods yields the best curvature estimates according to [WS93]. It is, however, important to notice that the three different methods reviewed in this paper differ mainly in the way that the orientation is estimated and in how the differentiation step is performed. The only constraints that seems to have been followed is that there is only one smoothing operation. By relaxing this constraint it is very easy to see that if we use a our generalized definition of path based method twice (once for the orientation estimation and once

for the computation of the derivative of θ), then the only difference between orientation based methods reformulated in this way and path based methods is in the way the derivation and smoothing steps are combined.

As path based methods have the full generality we are looking for, whilst keeping the possibility of expressing the derivative computation in the way that gives the best results for curvature (orientation based methods), it is the kind of method we have chosen to use.

5.2 General Outline of the Method

We now give a more precise description of our slightly modified path based method.

Suppose that some quantity Φ is defined along the edge (for example, the edge point coordinates or the edge orientation). These quantities define, along with the ordered set of points that represent the edge, a function along the edge. This function can be parameterized, and in theory, any parameterization can be used as the formula we will use for computing the derivative values are invariant to re-parameterization. Practically, however, two special kind of parameterizations play an important role and the quality of the results strongly depend on the chosen parameter [SA85, WS93]. The first way to define such a quantity is to use the index of the point in the edge as the parameter (this is used frequently as it is very easy to compute). The second kind of parameterizations are those based on an estimation of an arclength (this parameterization can be an Euclidean, an affine or a projective arclength: there is not yet much experience on how to compute the last two quantities, and so we adopt Euclidean arclength for the remaining of the discussion). The importance of a proper estimation of the arclength must not be overlooked as such a quantity defines how the 2D Euclidean structure of the image is mapped onto the curve (but we delay this discussion till the next chapter). For now, let us just assume that a parameter s is associated to each of the points of the curve.

Around each of the edge points $\mathbf{m}$ (except perhaps around the points that lie on the sides), it is possible to define a set $\mathcal{V}_{\mathbf{m}}$ of neighboring points. This can be done in many different ways, three examples are:

- Taking a given number of points on each side of $\mathbf{m}$ along the curve (i.e. in the order defined by the edge). This parameterization is important as it gives neighborhoods that contain a fixed number of points: this privileges the number of samples over the uniformity of the size of the neighborhoods along the curve.

- Taking the points that are within a disc of fixed radius and centered at $\mathbf{m}$. This sampling scheme respects the uniformity of the size of the neighborhoods but makes no use of the curve structure.

- Using the parameter s as a distance function along the curve and selecting the points that are within a certain distance of $\mathbf{m}$. In contrast to the first parameterizing scheme, this privileges the uniformity of the size of the neighborhoods along the curve over the sampling of the curve.

Notice that varying the size of the neighborhoods allows some kind of scale space computation. Such an implementation may be more difficult to implement but it results in better computational times, since only one-dimensional data sets are used. The exact relation of such "scale space" computation with the standard scale space is, however, not obvious.

Independently, in each neighborhood $\mathcal{V}_{\mathbf{m}}$, a local model is fitted to the data Φ considered as a smooth function of s. This yields a continuous estimate $\hat{\Phi}(s)$ of Φ in the neighborhood $\mathcal{V}_{\mathbf{m}}$. This local model can then be differentiated in order to obtain estimates of Φ_s, Φ_{ss}, ... of the derivatives of Φ with respect to s. These define new quantities along the curve that can be combined to obtain the looked for derivatives (see for example the formulae given in section 2.4). Eventually, these new quantities can again be differentiated using the above method.

The computational scheme can thus be summarized as:

1. Compute a parameterization s along the curve. Any origin can be taken.
2. Choose a quantity Φ defined along the curve that has to be differentiated.
3. Select a neighborhood $\mathcal{V}_{\mathbf{m}}$ around each point $\mathbf{m}$ of the edge.
4. Considering Φ as a smooth function of s in the neighborhood $\mathcal{V}_{\mathbf{m}}$, fit a smooth function $\hat{\Phi}(s)$ to the data points (s, Φ) that are in $\mathcal{V}_{\mathbf{m}}$.
5. Take the derivative of $\hat{\Phi}$ at $s_{\mathbf{m}}$ (the parameter value corresponding to $\mathbf{m}$). This defines a new quantity defined along the curve that can be combined with other ones. The result of such a combination can, in turn, be differentiated starting at step 1.

Remark 5.1 *It is important to notice that the derivative procedure described in this section leads to completely independent computations for the different points along the curve. Thus, if speed is an important factor all the steps involved in this computation (except maybe for the neighborhood computation) can be implemented very efficiently on a SIMD parallel machine. This is not the case for some standard filtering techniques such as recursive filtering.*

Notice that, till now, we have not specified the exact fitting procedure because such a choice is not relevant for the general description and any fitting procedure gives rise to an estimator (with varying accuracies however). The following section presents the technique we have chosen to use for this procedure. An attempt to explain why this choice is better suited for derivative computation is presented.

5.3 Chebyshev Polynomials

The purpose of this section is to describe a method that allows the fitting of polynomial curves to data in a simple and efficient way. It is based on Chebyshev polynomials: these polynomials have a lot of very interesting properties described in the mathematical literature. Only the relevant details for our discussion are given here, most of the details can be found in [PFTV88] and [FP68].

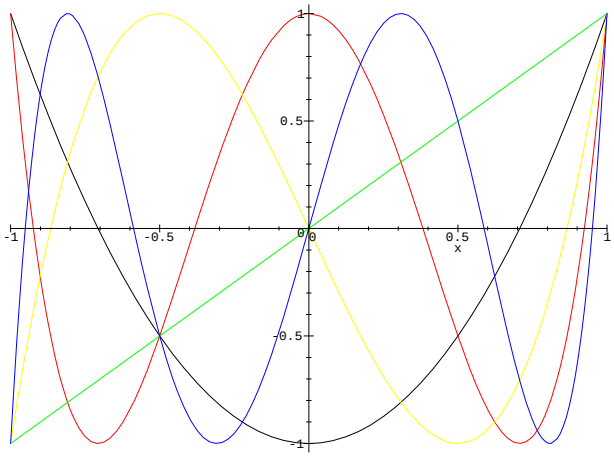

Figure 5.1: Chebyshev polynomials T_0 through T_6. Notice how T_n has exactly n real zeros that are all in the interval $[-1, 1]$ and how at all maxima $T_n(x) = 1$, while at all the minima $T_n(x) = -1$, so that the T_n polynomial is bounded between ± 1.

5.3.1 Mathematical Properties

The Chebyshev polynomials constitute a family of polynomials that can be indexed by their degree. The Chebyshev polynomial of degree n is denoted T_n and is defined in the interval $[-1, 1]$ by the formula:

$$T_n(x) = cos\left(n\ cos^{-1}(x)\right) \ .$$

Although, it does not look like a polynomial, it is actually one. The T_n family can be defined by the recurrence formulae:

$$\begin{aligned} T_0(x) &= 1\ , \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x)\ . \end{aligned}$$

Figure 5.1 displays the graph of the first 6 Chebyshev polynomials.

Moreover, Chebyshev polynomials constitute an orthogonal polynomial basis with respect to the special scalar product:

$$\int_{-1}^{1} \frac{f(x)g(x)}{\sqrt{1-x^2}} dx \ .$$

There is also a discrete analogue of this scalar product:

$$\sum_{k=1}^{m} T_i(x_k)T_j(x_k) = \begin{cases} 0 & i \neq j \\ m/2 & i = j \neq 0 \\ m & i = j = 0 \end{cases} \tag{5.1}$$

where the x_k $(k = 1 \ldots m)$ are the m zeros of the polynomial T_m $(x_k = cos\left(\frac{\pi(k-\frac{1}{2})}{n}\right))$. It is easy to verify that this formula yields the following theorem:

Theorem 5.1 *Suppose that $f(x)$ is an arbitrary function defined in the interval $[-1, 1]$. Defining the N coefficients c_i $(i = 0 \ldots N-1)$ by:*

$$c_i = \frac{2}{N} \sum_{k=1}^{N} f(x_k)T_i(x_k) \ .$$

Then, the approximation formula

$$f(x) \simeq -\frac{1}{2}c_0 + \sum_{k=1}^{N} c_k T_k(x) \ , \tag{5.2}$$

is exact *for x being equal to the N zeros of $T_N(x)$.*

For a fixed N, equation 5.2 provides a polynomial approximation of f. Once such a set of coefficient c_i have been computed, it is easy to evaluate the polynomial at a point using the Clenshaw's recurrence formula [PFTV88]. Similarly, the derivative of the approximation can be obtained very simply. If c'_i are the Chebyshev coefficients of the derivative, we have:

$$\begin{aligned} c'_m &= c'_{m-1} = 0 \ , \\ c'_{i-1} &= c'_{i+1} + 2c_i \text{ for } i = m-1 \ldots 1 \ . \end{aligned}$$

5.3.2 Using Chebyshev Polynomials in a Fitting Procedure

What makes the particular approximation method based on theorem 5.1 attractive is that it is possible to show that truncating the sum appearing in equation 5.2 to an order $m \ll N$ yields a polynomial of degree m that is very close to the minmax polynomial which approximates the data[2]. This minmax polynomial is defined as the polynomial of some given degree that minimizes the maximum error between the model and the data points. This minimization of the maximum error is an interesting property as it looks pretty much like the mathematical notion of uniform convergence which is of great interest when it comes to differentiation. An algorithm exists for the computation of this minmax polynomial (the Remez algorithm), but it is very expensive. So, we keep the cheap approximation based on Chebyshev polynomials as the results obtained by the two different methods are very close. Another way to look at this truncation operation is that there is a close relation between the Chebyshev polynomials and the discrete Fourier transform. Keeping the coefficients of lower degree (up to the bound m) is somehow equivalent to smoothing out high frequency components.

This is the main reason for using this method rather than any other standard filtering technique. Two other reasons are:

- Because the weighting function give more importance to the sides than to the center of the data set, there are far fewer side effects when using Chebyshev polynomials than when using standard (i.e. exponential filtering). Since in our case, we will deal with small sets of data (typically from 5 to 80 points) and since we will smooth then quite a lot, standard filtering techniques will usually lead to significant side effects that will at least partially invalidate the results.
- We want a method that is able to work with non uniform samples. Of course, this still can be done with the standard filtering techniques but it involves a re-sampling of the data.

The main problem which arises when implementing the previous scheme is that we need to be able to compute values at the points x_k for the function we want to

[2] This result is mainly based on the fact that Chebyshev polynomials are bounded between ± 1 as this property allows for a computation of an upper bound on the error made with the approximation when ignoring some terms of the sum (see [FP68] a complete proof of this property). [PFTV88] gives an interesting discussion about Chebyshev polynomial approximation.

approximate. Since the sampling is usually fixed, there is no control on the abscissa of the measured data. The solution we have adopted is simply to use the piecewise linear function that goes through all the measures as input. A standard mathematical result shows that when the resolution increases, doing such an approximation converges towards the curve. Notice however that, in doing so, the parameter defined on the curve plays an important role.

Remark 5.2 *In all the subsequent uses of the method described in this section, the parameter m has been fixed to 5 and the only tuning parameter is the size of the neighborhood. This value has been established by experience but it is interesting to notice that two independent tunings of this parameter have led to this same value as "the best optimum".*

5.4 Methodology

Testing the quality of the computed values for the spatio-temporal parameters is not an easy task. Since we plan to use these values quantitatively in a minimization scheme, there is a strong need for having a precise quantitative measure of the errors. On the other hand, we need to test the quality of the results in a situation that is as close as possible to the "real" set-up that will be used in the final application. Two different experimental protocols, each of which having strengths and weaknesses, are proposed in the following sections. The goal of each of these is to provide some reference values that will be compared with those that we obtain. Such reference values must be either theoretical values or else good approximation of them obtained, for example, by using a robust method involving some a priori knowledge about the observed curve.

5.4.1 Experimental Protocol 1

In computer vision, just like in many other fields, certain models are assumed which allow us to study a given problem in a precise and quantitative way. Such models include the pinhole camera or the step-edge models. Just like in any other science, we must be aware of the fact that these models are only approximations and that the results hold as long as the models are accurate enough for the situation that we are looking at. Most of the models which are used are mathematically well founded, there are however some phenomena that are more difficult to model: the quantization

of input data and the errors introduced in some of the treatments that are applied to the images (such as edge extraction) are particularly difficult from that point of view. The noise introduced by these steps is however quite important and it is necessary to have a tool which estimates the effects it has on a given computation.

One obvious way to generate data corrupted by these noise effects is to work with synthetic images. However, the problem we have to face with is slightly more complex due to the dynamical nature of sequences of images: it is necessary thus to have a 3D mathematical model that is moved from one frame to an another and projected, using computer algebra, onto a camera for each of these. The obtained formulae are then used to:

- Generate the quantized image of the curve for each time instant. Noise can, eventually, be added to the intensities. All the images constitute a sequence representing the motion of the 3D curve.
- Compute the theoretical values of the spatio-temporal parameters as functions of time and of a spatial parameter describing the position of a point onto the curve.

The edges are extracted using standard methods from the images , the spatio-temporal parameters are computed, and the results obtained are compared to the theoretical values computed using the theoretical data. Figure 5.2 schematizes the whole process.

Remark 5.3 *The theoretical formula obtained for the spatio-temporal derivatives are huge! Thus, we have used automatic tools to generate the C code (complete C functions) corresponding to the previous formula from the description available under the computer algebra system. These tools involve a C code generator that we have adapted from the one of Maple (see appendix F for a complete description) and many small hand-made tools that simplify the result by detecting the common sub-expressions in the formulae. If such an approach had not been used, the size of the resulting file would have been 44 Kbytes. This allows us to appreciate the amount of work that would have been necessary if we had not made use of the software tools. Moreover, such an approach simplifies the testing process as it virtually suppresses the possibility of a coding error while transcribing the formulae.*

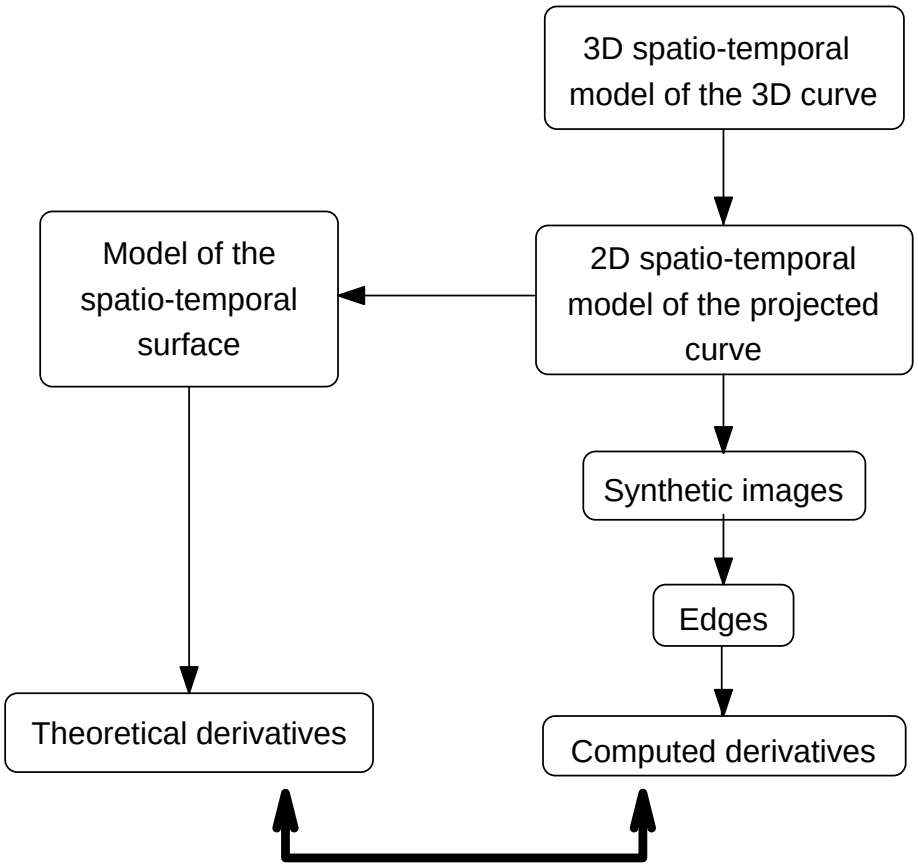

Figure 5.2: Experimental protocol 1: a 3D model of the curve is used to compute a sequence of images as well as the theoretical spatio-temporal parameters associated to it.

The scheme described here models quite well the problems of discretization, intensity noise and edge detector generated noise. Figure 5.3 shows four images excerpted from such a sequence.

5.4.2 Experimental Protocol 2

The main problem with the previous approach is that it is difficult to model some non-linear behaviors and some noise effects that may affect a real acquisition system. For example, lens distortion and digital-analogic conversions of the video signal are sources of noise that are not easy to cope with. Moreover, some of these may vary with the hardware used or with time. The process of image generation is actually a very complex one and it is difficult to model it without obtaining an intractable result. Thus, without an huge effort of modelization (which is not the topic here), the best way to obtain such data is to use real images. Since there is no easy means to obtain some mathematical results directly in this case, the best that can be done is to use some a priori knowledge about the observed curve to estimate some very good measured values. We have made use of the fact that the image of a 3D conic is always a conic (up to non-linear distortions that we will neglect in first approximation but that may very well have an influence on the results). It is then possible to fit, at each time instant, globally to the data points of the curve a conic model and then to use this model to measure the spatial derivatives. As it was stated previously, such a global method leads to very good estimates of the derivatives. Of course, it is possible to choose other types of 3D curves such as lines or cubics, but conics are the simplest model for which the derivatives we are interested in are non-trivial. The fitting procedure we have used is based on the minimization of the Euclidean distance of the data points to the conic model. Thus for each time instant, we have the set of the six coefficients of the quadratic form defining the conic.

Coping with time derivatives is slightly more difficult since we know nothing about the image curve motion. Actually the 3D curve plane is on the top of a rotating table and thus the 3D motion is known. It is however quite difficult, with the used experimental set-up, to have an accurate estimate of the relative position of the camera to the curve. To get rid of this problem, we have used the estimator described in the previous sections (actually the method was first designed for this particular task): the idea is to consider each coefficient of the quadratic form defining the conic as a function of time, for each of these functions the method can be applied and yields values for the first and second order time derivatives of the corresponding coefficient.

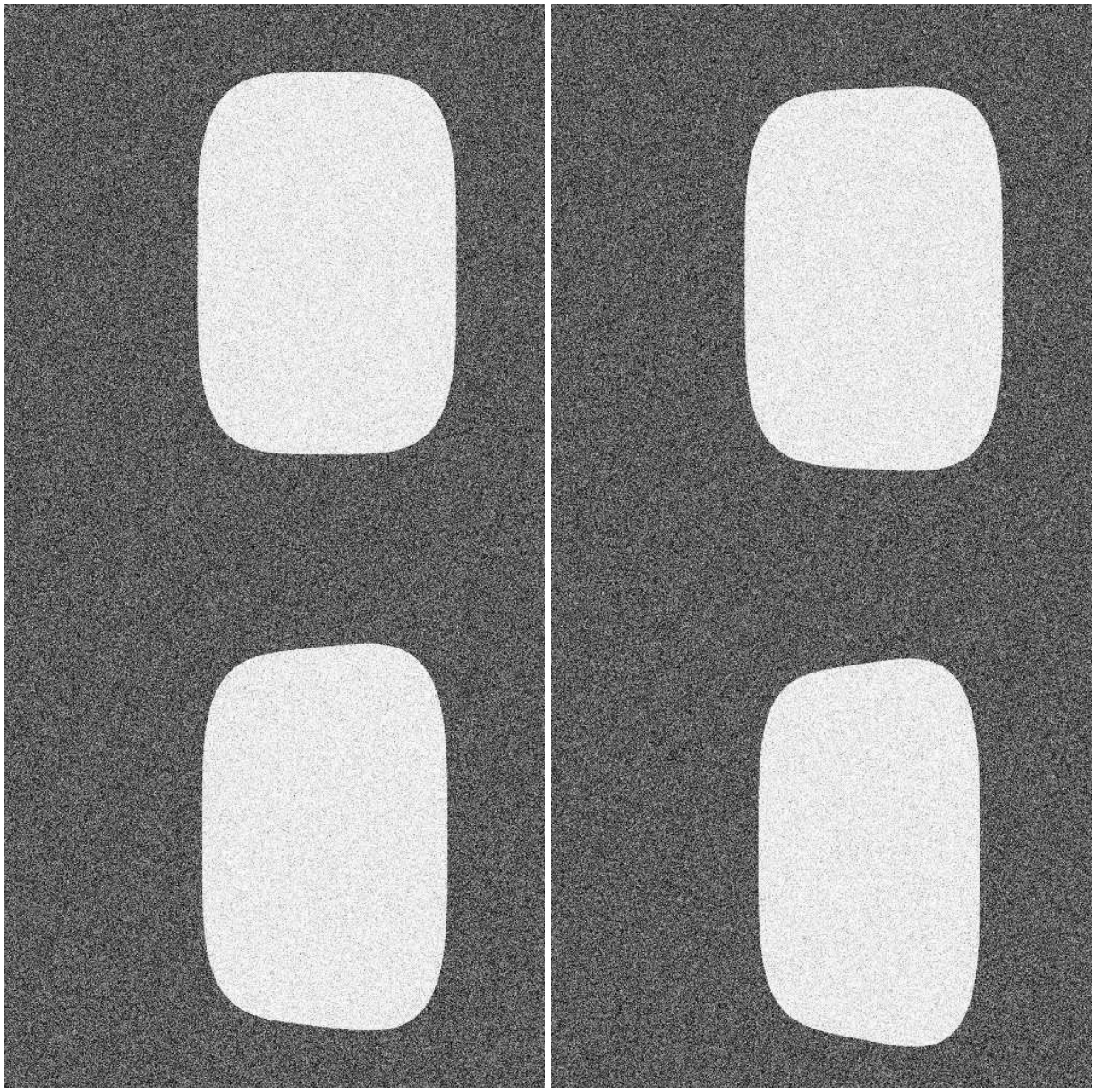

Figure 5.3: From left to right and from top to bottom: images excerpted from a synthetic sequence at times 0, 10, 20 and 30. A Gaussian noise of signal-to-noise ratio of 20% has been added on the intensity values.

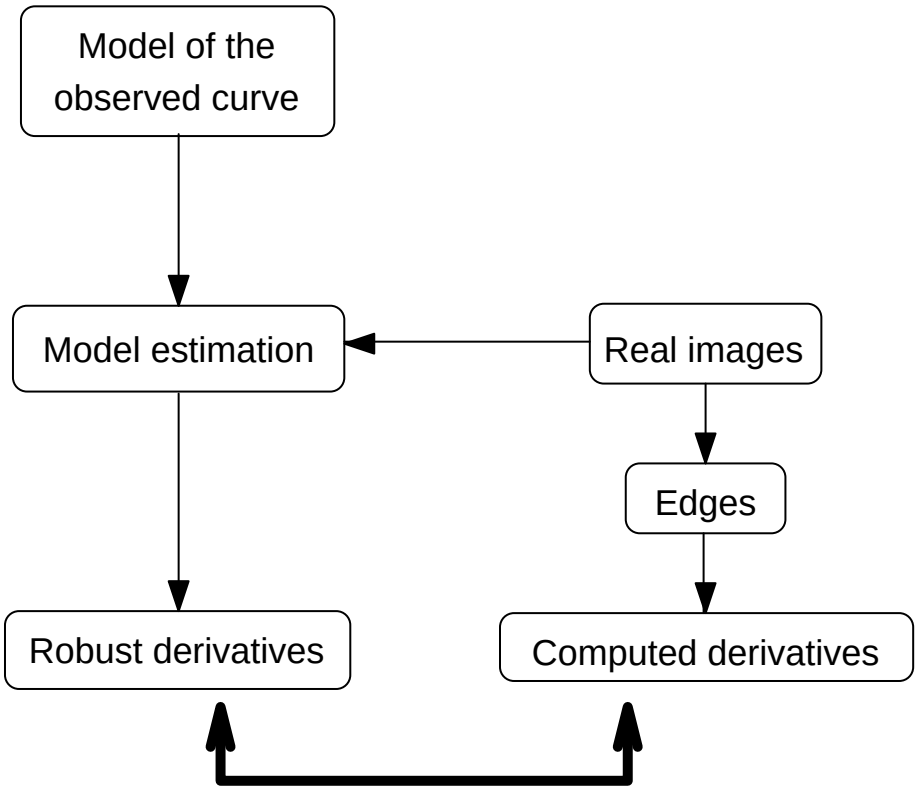

Figure 5.4: Experimental protocol 2: a 3D curve is chosen such that the type of its observed projection is known. This model allows an accurate estimation of the spatial derivatives whereas the temporal ones are more prone to errors since there is no model for time deformation of the observed curve.

These values, along with the original coefficient and the x and y coordinates of a point belonging to the edge, can then be used to compute the derivatives that interest us all along the edge (the formula that express the values of the derivatives in terms of the time and space partial derivatives of the quadratic form are given in chapter 10). Note again that, the reference model of the spatio-temporal surface depends on 2 parameters: a spatial one and a temporal one. Figure 5.4 summarizes the whole set-up. Figure 5.5 show four images excerpted from such a real sequence of conics.

Remark 5.4

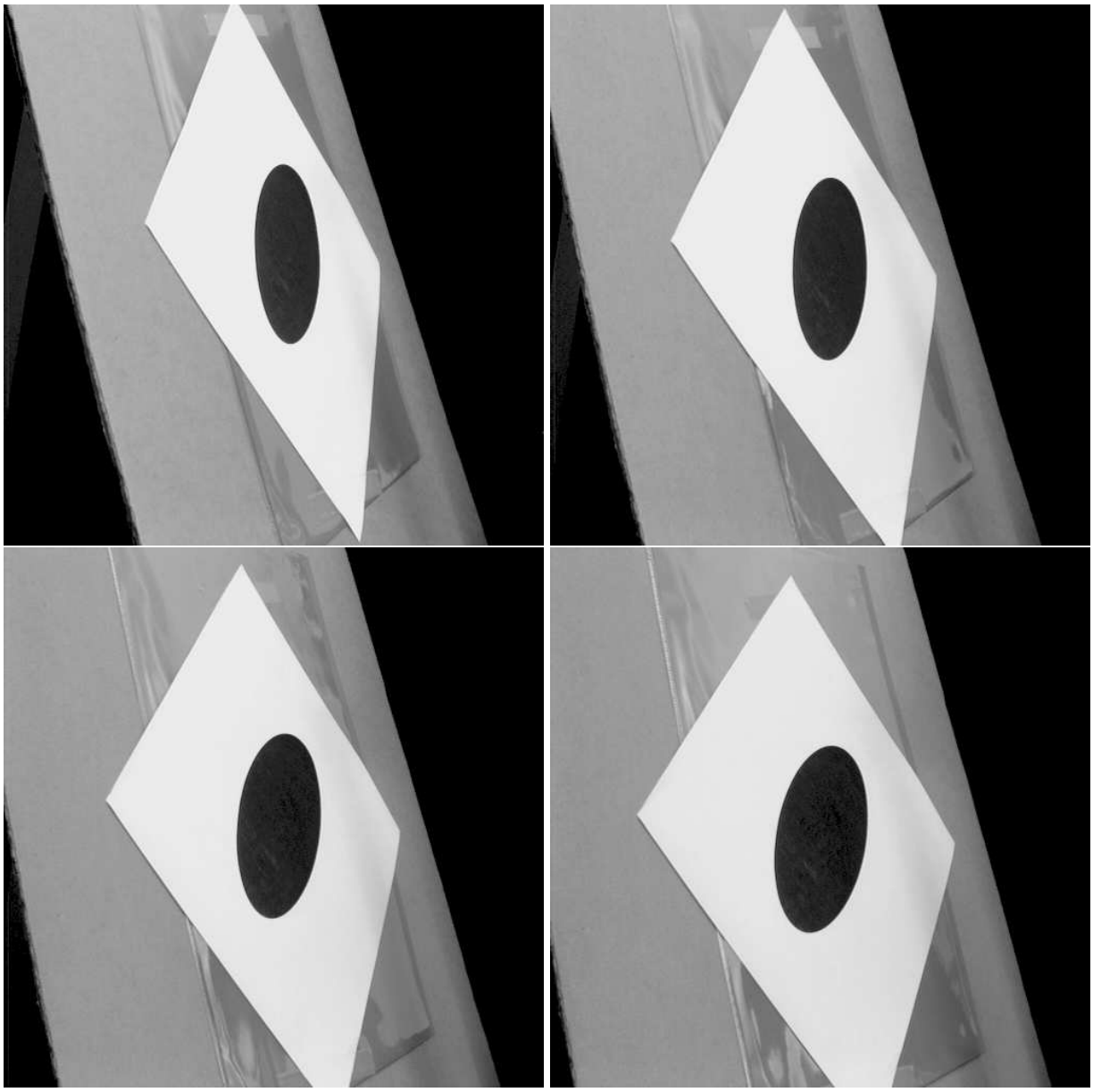

Figure 5.5: From left to right and from top to bottom: images excerpted from a real sequence of conics at times 0, 10, 20 and 29.

- *There is obviously an* experimental protocol 0 *that takes perfect data as input. Obtaining such data, in our case, is easily done using the tools developed for experimental protocol 1. Sometimes, such a step is so trivial that it can be ignored (this is indeed quite the case for the spatio-temporal derivative calculation), however, obtaining a method that works even with perfect data is not always an easy task. This step is then a good choice to start with. For example, it has proved to be very useful with the motion computation described in part III.*

- *Even if the two proposed methods provide a very effective way to estimate the quality of the results, they either fail to model some of the noises that corrupt images or they do not provide very reliable second order temporal derivatives. It would certainly be very interesting to develop, if possible, a very complete model of image formation that will allow to generate more realistic images (the term realistic means that the images would have the same noise corruption as real images), since this will give images for which exact theoretical values are known and for which it will be possible to isolate and quantify the effects of each noise source separately.*

- *Approaches related to the ones proposed here have been used for estimating the accuracy of optical flow methods (mainly for point based methods). See [BFB94] and [ON94].*

5.4.3 Comparing References and Measured Values

Matching the Edge Points and the Model

In order to be able to compare the reference values against the measured ones, one problem that remains to be solved with these two methods is: given some measured point on the spatio-temporal surface, find the corresponding point in the reference model (i.e. the spatial and temporal parameters giving that point). Finding the time parameter is easy since we suppose that the images in the sequence are sampled regularly in time (even with real acquisition systems, this seems to be a sensible hypothesis).

The spatial parameter is slightly more difficult to obtain. However, assuming that the measured point is not far from the theoretical curve, that the parameter describes the curve quite uniformly (this means that the derivative of the arclength

with respect to the spatial parameter does not vary too much along the curve), and that the scale of the curve is known a priori, it is possible to design an efficient method that works quite well in practice. Suppose that the curve is parameterized at a given time instant by the parameter μ. Let us call $\mathcal{D}$ the range in which this parameter must vary to describe all the points of the image curve and $\mathbf{m}_0$ the edge point that we try to locate.

1. The curve is cut out into N pieces by dividing the range $\mathcal{D}$ into N intervals of equal length. N is determined such that, in each piece, the curve does not have many details i.e. behaves grossly like a straight line. The scale and the uniformity hypothesis ensure that this can be done. Practically, for the images shown in this chapter the value $N = 50$ have always been sufficient.

2. For each piece of curve, the distance of the point $\mathbf{m}_0$ to the border points is computed. The two pieces of curve that are limited by the border point for which this distance is minimal constitute a gross estimate of the location of $\mathbf{m}_0$ onto the model.

3. This estimate is then refined by using a dichotomic method for which the size of the piece of the curve is divided by two at each step. The process is stopped when this size decreased below some fixed threshold that controls the accuracy of the localization onto the model or when the distance has reached a minimum value. Provided that N is big enough and that $\mathbf{m}_0$ is close enough to the model curve (which is always the case in the experiments proposed here) this dichotomic scheme converges towards the point of the model curve that is the closest to $\mathbf{m}_0$.

Remark 5.5 *Of course, the algorithm we have described would be untrue if any of the hypotheses we have made are false and especially if the point* $\mathbf{m}_0$ *is too far from the model. Fortunately, as the model we have is already quite accurate, this does not happen with our setup. Moreover, since it is used only for the validation of the algorithm, we can afford to spend some time for tuning the parameters of this localization in order to get the best possible mapping between the edge points and the model.*

Error Measurements

In the forthcoming chapter, we present results obtained with various methods that compute derivatives. These results are systematically compared to the best reference values that we have access to in a quantitative way. However, apart from the error curves that will be shown, it is interesting to be able to characterize the quality of an approach globally (for all the curve), using a small number of quantitative parameters. Noting $x(s)$ the reference value function for the estimated quantity and $e(s)$ the error function corresponding to the estimation, we define two such quantities.

- Mathematically, the most common way to define the distance between two curves is to take the maximum over all the points of the absolute value of the error. This leads to the criterion:

$$error_{max} = max_{\text{over all the points of the curve}} |e| \ .$$

- The main fault with this first criterion is that it is well-known that in most cases a relative error is better suited to appreciate the quality of an estimator than an absolute one. Unfortunately, this error is not defined when an estimate corresponds to a zero value. Most most of the derivative curves that will be shown exhibit zero crossings at which the relative error will go to infinity. So, taking the maximum value over all the points of the curve of the absolute value of the relative error is not a good measure to adopt. To obtain a relative global measure of the quality of an estimator, we have thus divided the mean over all the points of the absolute value of the error by the mean value of the absolute value of the model values. This gives us a second criterion that we call the global error:

$$error_{glob} = \frac{mean_{\text{over all the points of the curve}} |e|}{mean_{\text{over all the points of the curve}} |x|} \ .$$

5.5 Conclusion

In this chapter, we have presented the method that we propose to use for estimating the derivatives and defined the experimental protocols we have used to validate it.

These methods will also be used to compare the method we propose to some others that can be found in the literature or that we have tried before ending with the one that is proposed in this chapter.

From the theoretical point of view, the main strengths of the tools proposed in this chapter are:

For the derivative estimator:

- The method is very general and is not restricted to the computation of some particular derivatives (as for example most of the methods described in [WS93]).
- It can be used with non uniformly sampled data.
- If speed is needed, it can be implemented quite easily on a parallel computer.

For the methodology:

- Very good reference values have been obtained with synthetic images. These allow for an accurate measurement of the errors arising from discretization and edge extraction.
- Real images have been used to obtain reference values. However, the temporal derivatives obtained with these might suffer from the difficulty to establish a temporal model of the evolution of the template curve. Yet, we believe that the reference values obtained this way are good enough to characterize the quality of the computed derivatives.

The main drawbacks are of course the speed of the computation of the derivatives (it is certainly not real time yet) and the lack of a more accurate way to obtain reference values with real images.

The details of the practical implementation and the accuracy of the obtained results are discussed in the next chapter.

Chapter 6

Experimental Results and Comments

This chapter gives all the details of our implementation of the derivative estimators we have used and presents the results that have been obtained with these. Emphasis is placed on the comparison of the obtained results with the reference values calculated as described in the previous chapter. However, in the case of curvature, we also compare our results with a standard image based approach. This chapter is organized as follows: first, we discuss the proper coordinate system in which the computation must be realized, then the problems of arclength estimation and of the construction of an efficient spatio-temporal structure allowing an easy retrieval of the neighbors of a given point. Then, the results that have been obtained for the different quantities are presented and a brief comment is made about the importance of starting with subpixelic edge coordinates.

Hereafter, we suppose that images are taken in such a rapid succession that the temporal continuity from image to image is approximately equal to the spatial continuity in an individual image [BBM87, BB89]. The basic underlying idea is that there is no reason to treat the spatial and temporal dimensions differently as far as

the estimation of only derivatives is concerned. For this reason, we have made the choice of working with long sequences of images.

6.1 Image Coordinates Versus Normalized Coordinates

As noted in chapter 3, the image and normalized coordinate systems defined on the retina are not equivalent for our purpose because they are related by an affine transform and since we are looking at Euclidean properties of the image. As shown in that chapter, even the most basic measure we need — the normal flow field β — is not well defined in the image coordinate system since orthogonality is *not* an affine property.

Moreover, it is not possible to compute the values we are looking for in the normalized coordinate system from those measured in the image coordinate system. Actually, such a transfer can be done only for $\mathbf{m}$, $\mathbf{n}$, β and κ. It is not possible to achieve such a computation with $\frac{\partial \beta}{\partial s}$ and $\partial_{\mathbf{n}_\beta}\beta$. The easiest way to convince oneself of this is to write the formulae relating the partial derivatives of an implicit function computed in the normalized coordinate system to those computed in the image coordinate system. From these formulae, it is possible to express all the quantities we are interested in terms of the partial derivatives obtained in the image coordinate system (these formulae can be found in section 10.1). Let us call $\mathbf{m}_n$, $\mathbf{n}_n$, β_n, κ_n, $\frac{\partial \beta}{\partial s}_n$ and $\partial_{\mathbf{n}_\beta}\beta_n$ the values obtained for the spatio-temporal derivatives in the normalized coordinate system (n stands for normalized). Now, it is also possible to compute these quantities in the image coordinate system. This yields the quantities $\mathbf{m}_i$, $\mathbf{n}_i$, β_i, κ_i, $\frac{\partial \beta}{\partial s}_i$ and $\partial_{\mathbf{n}_\beta}\beta_i$ (i for image). These quantities are in general different from their normalized correspondencies.

The dilemma is now the following: the image measures are easier to compute but it is the normalized ones that are of interest. Thus, it is natural to wonder whether it is possible to express the latter in terms of the former. Using the formulae described above, this is equivalent to eliminating the image partial derivatives appearing in the normalized quantities using the equations giving the image quantities in terms of the same image partial derivatives. Doing so proves that:

- $\mathbf{m}_n$ can be expressed as a function of $\mathbf{m}_i$. This is just the affine transform between image and normalized coordinates introduced in chapter 3:

$$x = h_1 u + h_2 v + h_3 ,$$

$$y = h_4 v + h_5 \ .$$

- $\mathbf{n}_n$ can be expressed as a function of $\mathbf{n}_i$:

$$\begin{aligned} \theta_n &= tan^{-1}\left(-\frac{h_1 \mathbf{n}_{iv} + h_2 \mathbf{n}_{iu}}{h_4 \mathbf{n}_{iu}}\right) \ , \\ \mathbf{n}_n &= \begin{bmatrix} cos(\theta_n) \\ sin(\theta_n) \end{bmatrix} \end{aligned}$$

- β_n can be expressed as a function of β_i and $\mathbf{n}_i$

$$\beta_n = -\beta_i \frac{ds_i}{ds_n} \ ,$$

$$\text{where } \frac{ds_i}{ds_n} = \frac{h_1 h_4}{\sqrt{(h_1 \mathbf{n}_{iv} - h_2 \mathbf{n}_{iu})^2 + h_4^2 \mathbf{n}_{iu}^2}} \ .$$

- κ_n can be expressed as a function of κ_i and $\mathbf{n}_i$.

$$\kappa_n = \frac{\kappa_i}{h_1^2 h_4^2} \left(\frac{ds_i}{ds_n}\right)^3 \ .$$

- The two other quantities ($\frac{\partial \beta}{\partial s\, n}$ and $\partial_{\mathbf{n}_\beta} \beta_n$) cannot be expressed as functions of the parameters $\mathbf{m}_i$, $\mathbf{n}_i$, β_i, κ_i, $\frac{\partial \beta}{\partial s\, i}$ and $\partial_{\mathbf{n}_\beta} \beta_i$.

This result has led us to working directly with normalized coordinates. In the remainder of this chapter (and hence in all the subsequent chapters), edge point coordinates have been expressed in the normalized coordinate system prior to any derivative computation.

Remark 6.1 *Experiments have been run in which $\beta = \beta_n$ was computed from the values of β_i and $\mathbf{n}_i$. This has shown that such a method does not exhibit the same error properties as the one proposed hereafter. The best obtainable quality is about the same but the amount of smoothing used to get it is different. This is not so surprising as the method we have sketched is* not *affine invariant.*

6.2 Computing the Arclength Parameter

As remarked in the previous chapter one of the main difficulties that arises with path methods is that it is firstly necessary to parameterize the curve in order to be able to define the concept of a quantity defined along the curve. As explained, we chose to parameterize the 2D curves by their Euclidean arclength. This is because such a parameterization defines a distance along the curve that preserves the Euclidean properties of the retina. Why have we chosen to preserve specifically these Euclidean properties? The answer is twofold:

- Euclidean arclength is much simpler to compute than affine or projective arclength. Yet, we show here that this quantity is not so easy to compute, and that care has to be taken in order to obtain good estimates for it.
- Our study of the properties of the spatio-temporal surface has been done in a Euclidean framework so that adopting Euclidean arclength is fully coherent with this study. Doing so is natural and suppresses more potential troubles than it introduces.

In full generality, since we want to cope with spatio-temporal surfaces, we should have worked with surfaces in the 3D spatio-temporal space which is considered as Euclidean. This would have led us to a two degree of freedom parameterization of the spatio-temporal surface. However, the study of chapter 4 has shown that a valid parameterization is (s, τ), where s denotes the Euclidean arclength, and τ denotes time. Adopting this special parameterization, we will show that, for the task of computing the spatio-temporal derivatives that interest us, it is always possible to restrict ourselves to 2D edges (i.e. curves).

The standard method for computing Euclidean arclength is to fix the arclength value of some arbitrary point of the curve and then, starting with this point as origin, accumulate the distance ds between successive points along the curve. However, such a method leads to significant biases even when the curve is a straight line. Taking points with integral coordinates lying on such a line, it is easy to see that the actual arclength is grossly over-estimated using this technique (see figure 6.1). A complete discussion about the estimation of line length can be found in [DS87]. Moreover, the bias that occurs is non uniform at small scales: the error between the computed ds and the true underlying value is not constant. With lines and at high scales, this bias

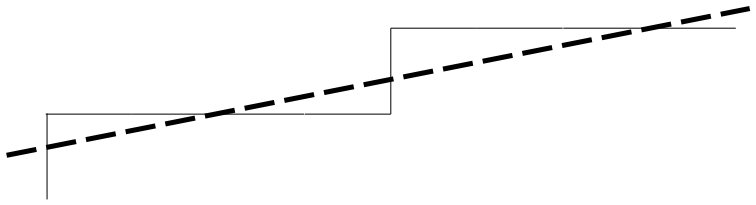

Figure 6.1: Comparison between continuous and discrete ds: the dashed line is the real line that has been approximated by the plain curve. The error we obtain here is 0.8 pixels for any length computed by taking five consecutive pixels.

becomes uniform, but there is no reason for this to be true when dealing with curves (basically the bias depends on the orientation of the line, and so as the orientation along a curve varies the bias becomes unpredictable). A nice way to see this effect is just by looking at the lengths of the curves: the result of this estimation gives overestimated results with a relative error of 10%. It is very interesting to notice that the theoretical bias with this method for straight lines is asymptotically equal to 6.6% [DS87]. Since this error is not uniform along the curve, there is no easy way to dispose of the bias. By plotting the error between the theoretic ds and the estimated one, it is possible to show that there is more error in highly curved parts than in parts that are almost straight lines.

Actually, digitization is the main source of noise that is responsible for this overestimation. So there are two ways to improve the estimation of the curvilinear abscissa:

- Use the curve-scale space a la [MM92] to smooth out the discretization effect. Figure 6.2 shows the evolution of the length of one curve from the synthetic sequence over scale space. As we can see, the length decreases rapidly at first, and then stabilizes to a value that is not far away from the real value (relative error of approximately 0.1%). Computing the scale space is slow, but the resulting accuracy is worthwhile. Moreover, computing the error between the theoretical ds and the estimated one after each scale-step computation, shows that the errors in ds is indeed decreasing.

- Another way to get rid of discretization effects is to obtain point coordinates that take real values. It is actually possible to obtain from discrete images edge points that are of subpixel accuracy ([TM84] for example). Figure 6.3

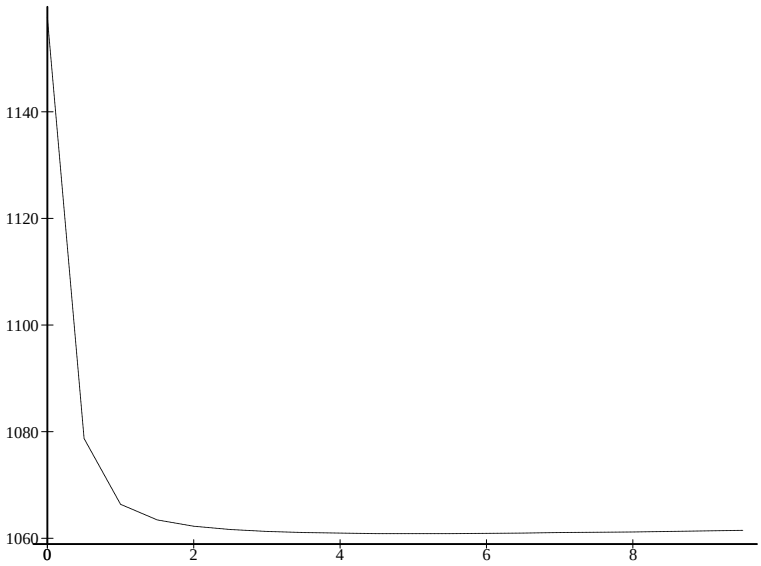

Figure 6.2: The evolution of the Euclidean length of a curve along scale space.

shows the quality of the computed arclength. It is interesting to note that our experimental setup shows that the accuracy of edge points is of about a fifth of a pixel with synthetic images and about half a pixel with real ones (see figure 6.4). Computing the length of the curve with these subpixel data as input gives estimates that are very close to the real results (at most a few hundredth of percent). This method will be preferred here since it gives comparable results at a low cost.

Remark 6.2

- *Note that if we combine these two results i.e. if we apply the scale space method with subpixel points as input, it should be possible to improve further the quality of the computed arclength but usually it is not worth the added cost.*
- *Since we will use formulae that are invariant to re-parameterization, one might ask why is it so important to have a good arclength parameter. This is because the standard mathematical expressions are not invariant with respect to all the re-parameterizations, but merely only to the ones that are related by a*

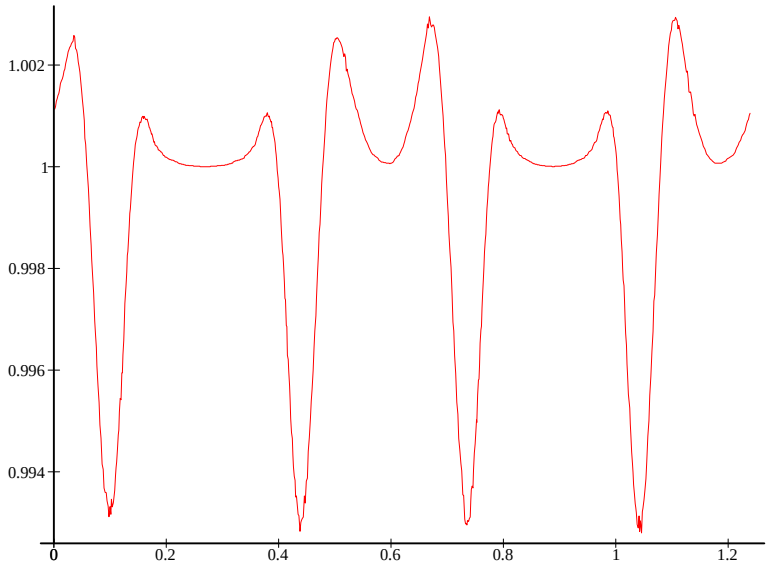

Figure 6.3: This plot shows the values of ds computed along the curve using our derivative computation. If this method and the computed arclength were perfect, the curve would have been constant and equal to one. The maximum error is thus of about 0.6%. This curve was obtained with the synthetic sequence.

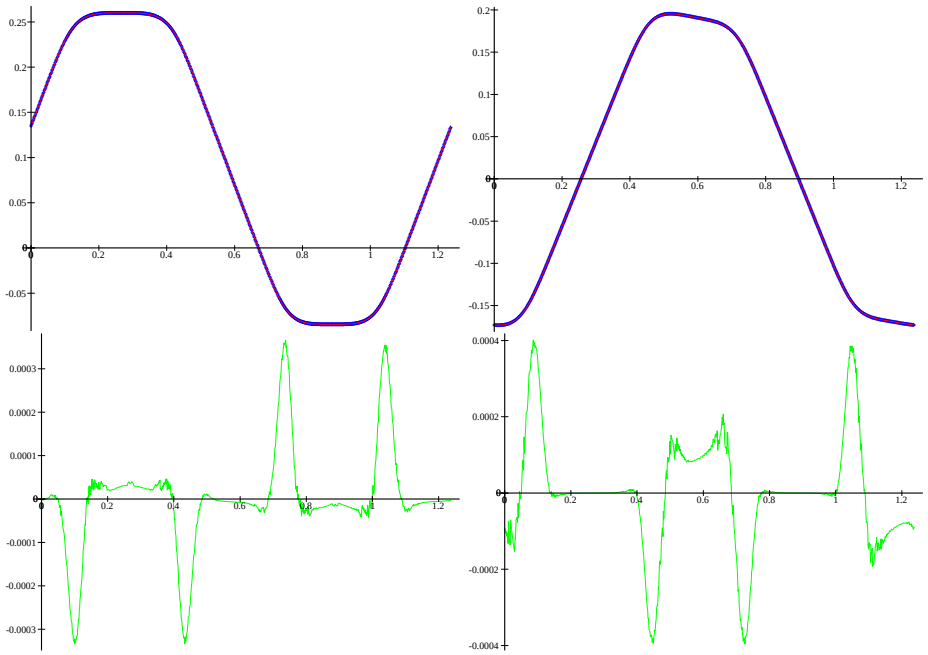

Figure 6.4: Top: the x (left) and y (right) functions along the synthetic curve. These two curves are actually the superpositions of a plain curves corresponding to the model and of a set of crosses representing the measured points: the matching between these two curve is so good that it is difficult to see the difference between these curves! Bottom: the error curves between the model and the measured values. Each error curve corresponds to the plot just above it. Notice that these plots are made in normalized coordinates. The worst error corresponds to an edge extraction accurate to a fifth of a pixel.

diffeomorphism. From this point of view, the most obvious parameterization, which is to index the points by their rank in the edge, is generally not equivalent to the Euclidean arclength. This has been shown for the example of circles in [WS93]

6.3 Representing and Building the Spatio-Temporal Surface

6.3.1 A Data Structure for the Spatio-Temporal Surface

In order to obtain good computation timings, it is necessary to gather the points constituting the spatio-temporal surface in a data structure that allows a fast retrieval of the neighbors of a given point. Moreover, looking carefully at the kind of neighbors we need to extract, we see that one is exclusively concerned with purely spatial neighbors (i.e. neighbor points along the observed curve at a given time instant) and with neighbors along the direction $\mathbf{n}_\beta$ in the tangent plane that we will call time neighbors in the sequel. To provide easy access, for each point we just stored the four "nearest" neighbors: two spatial and two temporal ones. "Nearest" spatial neighbors are easily inherited from the edge chains obtained after edge linking, for each point one of the pointer points to the previous point in the chain whereas the other one points to the subsequent point in the chain. If the pixel chain is open, null pointers are used for the undefined pointers in the first and the last point. Thus, the two spatial pointers organize each observed curve as a doubly linked list. Temporal neighbors are slightly more complex to obtain (especially when working with discrete data): since the direction $\mathbf{n}_\beta$ is not discrete, there is no unique solution for the "nearest" pixel in a given way along this direction. However, so long as our algorithms do not rely on such uniqueness, and use this "nearest" pixel approach as an easy access to time neighbors, it is possible to make the time "nearest" neighbor pointers to point to one of the nearest pixels for each way along direction $\mathbf{n}_\beta$. Thus, the structure of the time chains is not as simple as a doubly linked list: for example, due to inconsistencies in the normal vector $\mathbf{n}$ the time linking might not be reflexive. Figure 6.5 shows an example for the links associated with a point.

To obtain a complete spatio-temporal structure, we add to this set of four pointers an array of floating point values: there is one such value for each per-point quantity of interest; i.e. all the spatio-temporal parameters, but also some "administrative" quantities like the time t, the arclength s with some fixed origin, and